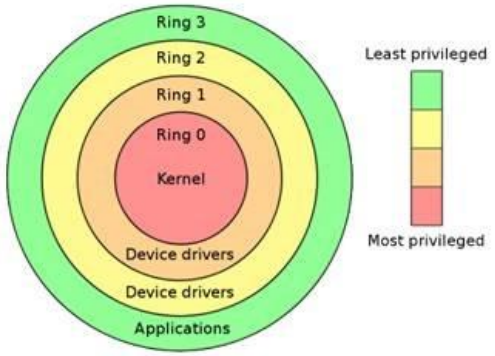
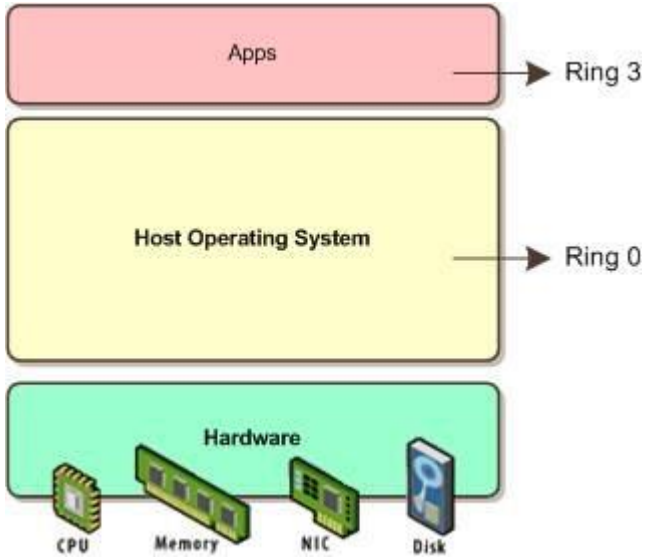
Qemu Libvirt & KVM

popsuper

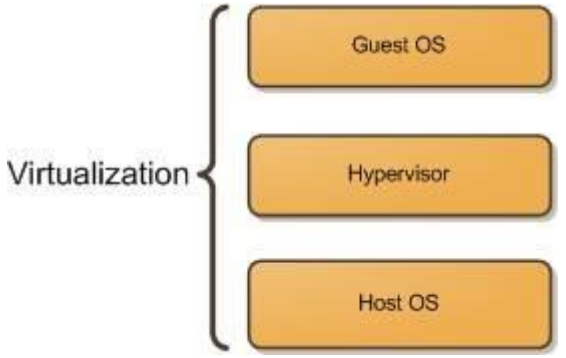
1.1 虚拟化的基本类型

- 无虚拟化
  - CPU一般设为四个Ring
  - Kernel Mode一般跑在Ring 0上
  - User Mode一般跑在Ring 3上
  - 对于一个普通的传统的Linux系统没有问题



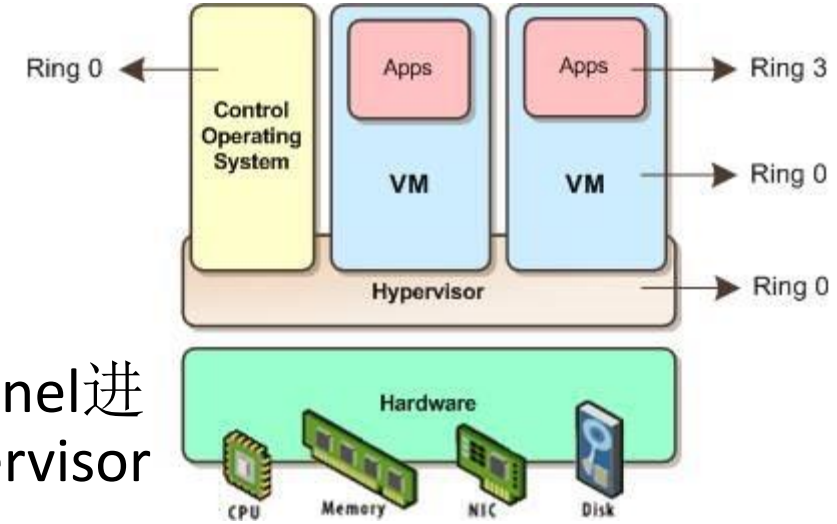


- 虚拟化
  - 在Guest机器和Host机器中间加一层Hypervisor
  - Host机器看它像跑在自己上面的程序
  - Guest机器看它像自己所运行的硬件
  - 如果Host机器和Guest机器都跑相同的Linux，它们的Kernel都想运行在Ring 0，可怎么办？



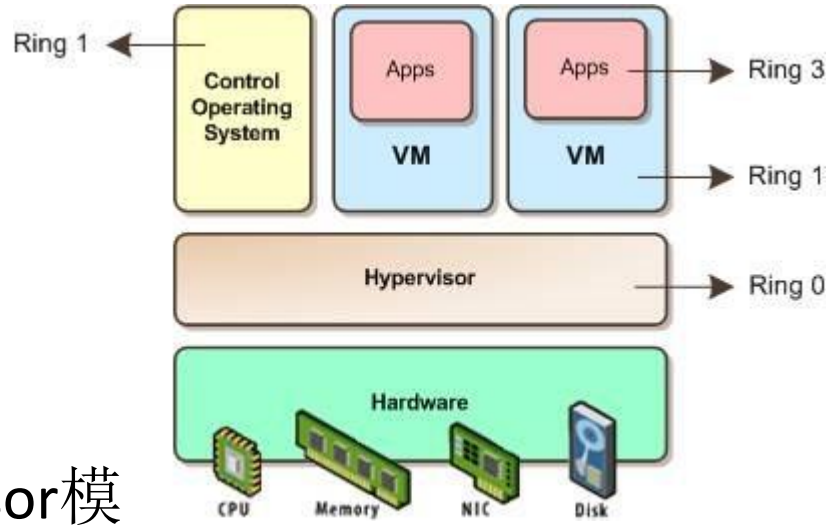
1.1 虚拟化的基本类型

- 半虚拟化Paravirtualization
  - Hypervisor运行在Kernel Mode, Ring 0
  - Guest OS不能直接运行在Ring 0，而是需要对Kernel进行修改，将运行在Ring 0上的指令转为调用Hypervisor
  - Guest OS上的APP运行在Ring 3



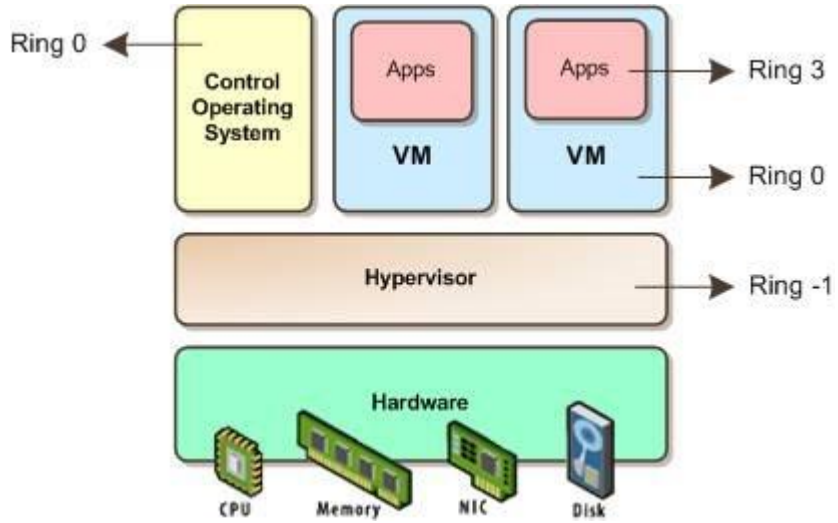
1.1 虚拟化的基本类型

- 非硬件辅助全虚拟化
- Full Virtualization without Hardware Assist
  - Hypervisor运行在Ring 0
  - Hypervisor对Guest OS提供CPU模拟，由Hypervisor模拟一个CPU给VM，VM不直接使用真实的CPU
  - Guest OS不做修改，还是试图运行在Ring 0上，只不过是模拟CPU的Ring 0
  - Hypervisor对Guest OS的Ring 0上的指令进行转译，变成真实CPU的指令，只能运行在Ring 1上



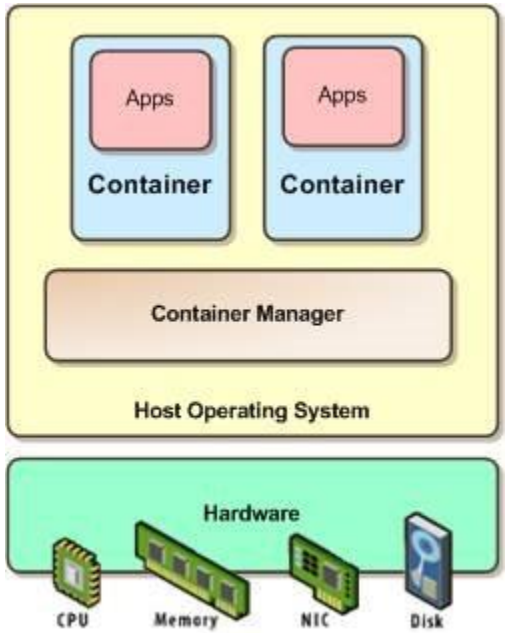
1.1 虚拟化的基本类型

- 硬件辅助全虚拟化
- Full Virtualization with Hardware Assist
  - Intel VT and AMD-V创建一个新的Ring -1单独给Hypervisor使用
  - Guest OS可以直接使用Ring 0而无需修改



1.1 虚拟化的基本类型

- OS virtualization



实验一：查看系统是否支持硬件辅助虚拟化

- 对于Intel CPU

grep "vmx" /proc/cpuinfo

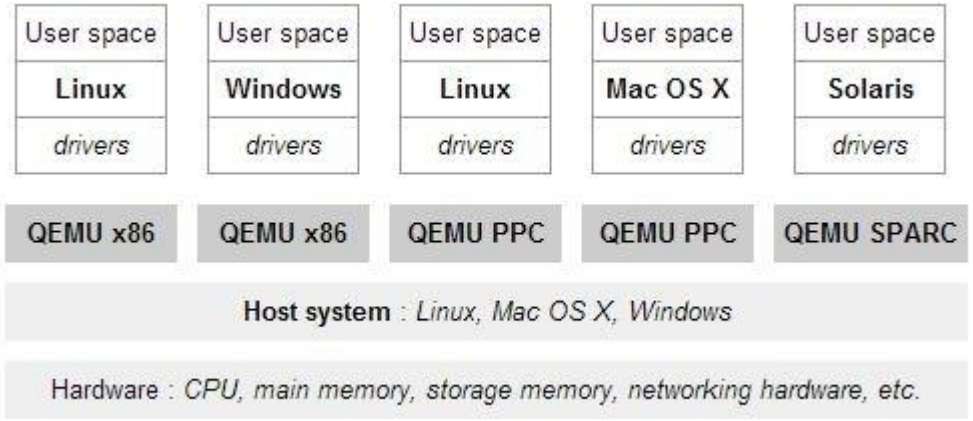
```
root@popsuper1982:/home/openstack# grep "vmx" /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
 nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est t
m2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 popcnt aes lahf_lm ida arat dtherm tpr_shadow vnmi flexpriority ept vpid
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
 nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est t
m2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 popcnt aes lahf_lm ida arat dtherm tpr_shadow vnmi flexpriority ept vpid
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
 nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est t
m2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 popcnt aes lahf_lm ida arat dtherm tpr_shadow vnmi flexpriority ept vpid
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
 nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est t
m2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 popcnt aes lahf_lm ida arat dtherm tpr_shadow vnmi flexpriority ept vpid
```

- 对于AMD CPU

grep "svm" /proc/cpuinfo

1.2 KVM Qemu Libvirt之间的关系

- Qemu是一个模拟器，它向Guest OS模拟CPU，也模拟其他的硬件
- 正如非硬件辅助全虚拟化讲到的，Guest OS认为自己和硬件直接打交道，其实是同Qemu模拟出来的硬件打交道，Qemu将这些指令转译给真正的硬件。
- 由于所有的指令都要从Qemu里面过一手，因而性能比较差



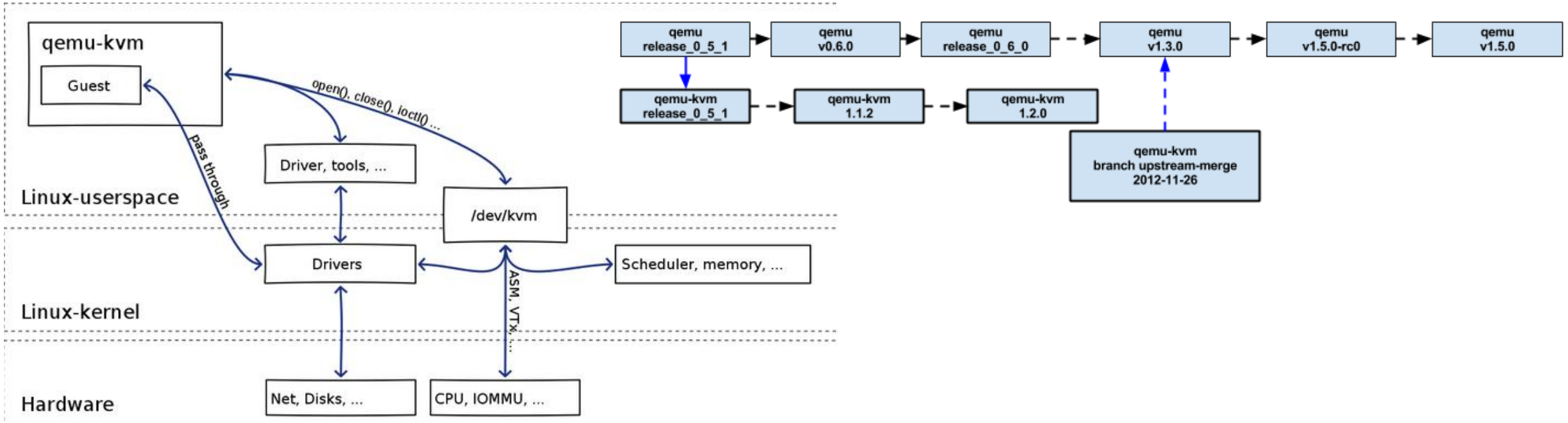
1.2 KVM Qemu Libvirt之间的关系

- KVM是内核的模块，采用硬件辅助虚拟化技术Intel-VT，AMD-V
- 使用KVM，Guest OS的CPU指令不用经过Qemu转译，直接运行，大大提高了速度
- KVM通过/dev/kvm暴露接口，用户态程序可以通过ioctl来访问这个接口

```
open("/dev/kvm")
ioctl(KVM_CREATE_VM)
ioctl(KVM_CREATE_VCPU)
for (;;) {
    ioctl(KVM_RUN)
    switch (exit_reason) {
    case KVM_EXIT_IO:  /* ... */
    case KVM_EXIT_HLT: /* ... */
    }
}
```

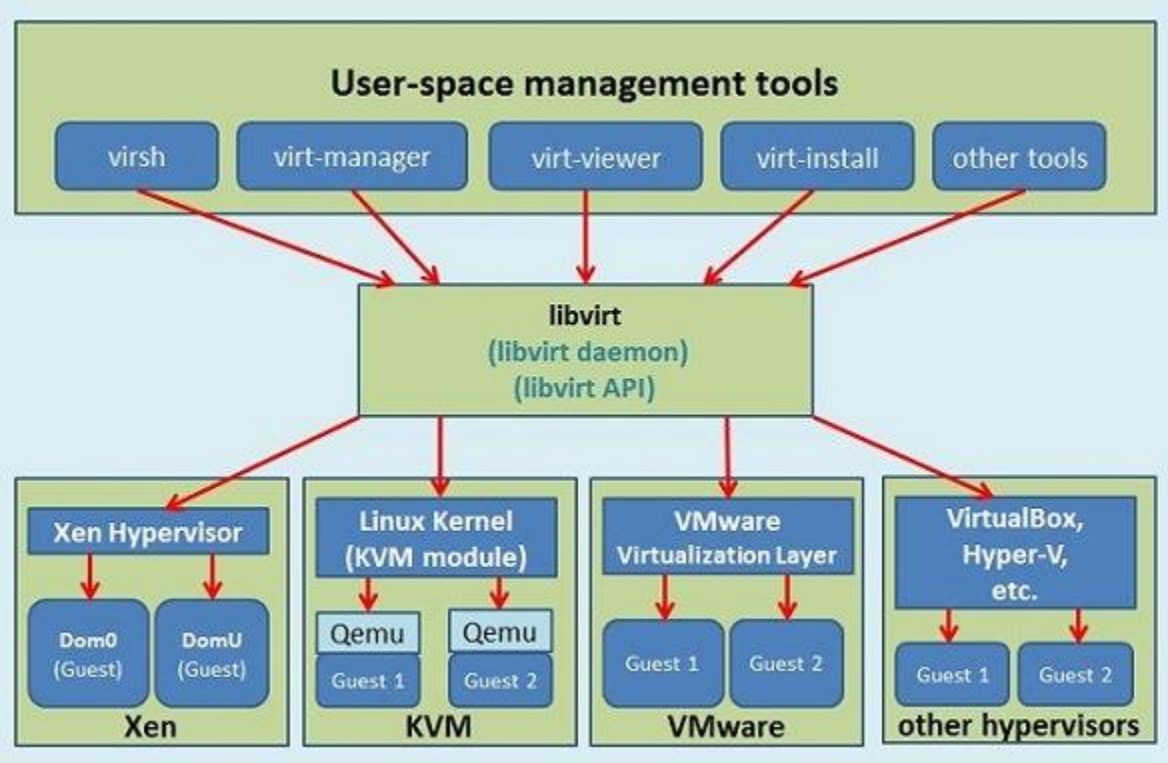
1.2 KVM Qemu Libvirt之间的关系

- Qemu将KVM整合进来，通过ioctl调用/dev/kvm接口，将有关CPU指令的部分交由内核模块来做，就是qemu-kvm (qemu-system-XXX)
- Qemu还会模拟其他的硬件，如Network, Disk，同样会影响这些设备的性能
- 于是产生pass through半虚拟化设备virtio_blk, virtio_net，提高设备性能
- Qemu-kvm对kvm的整合从release_0_5_1开始有branch，在1.3.0正式merge到master



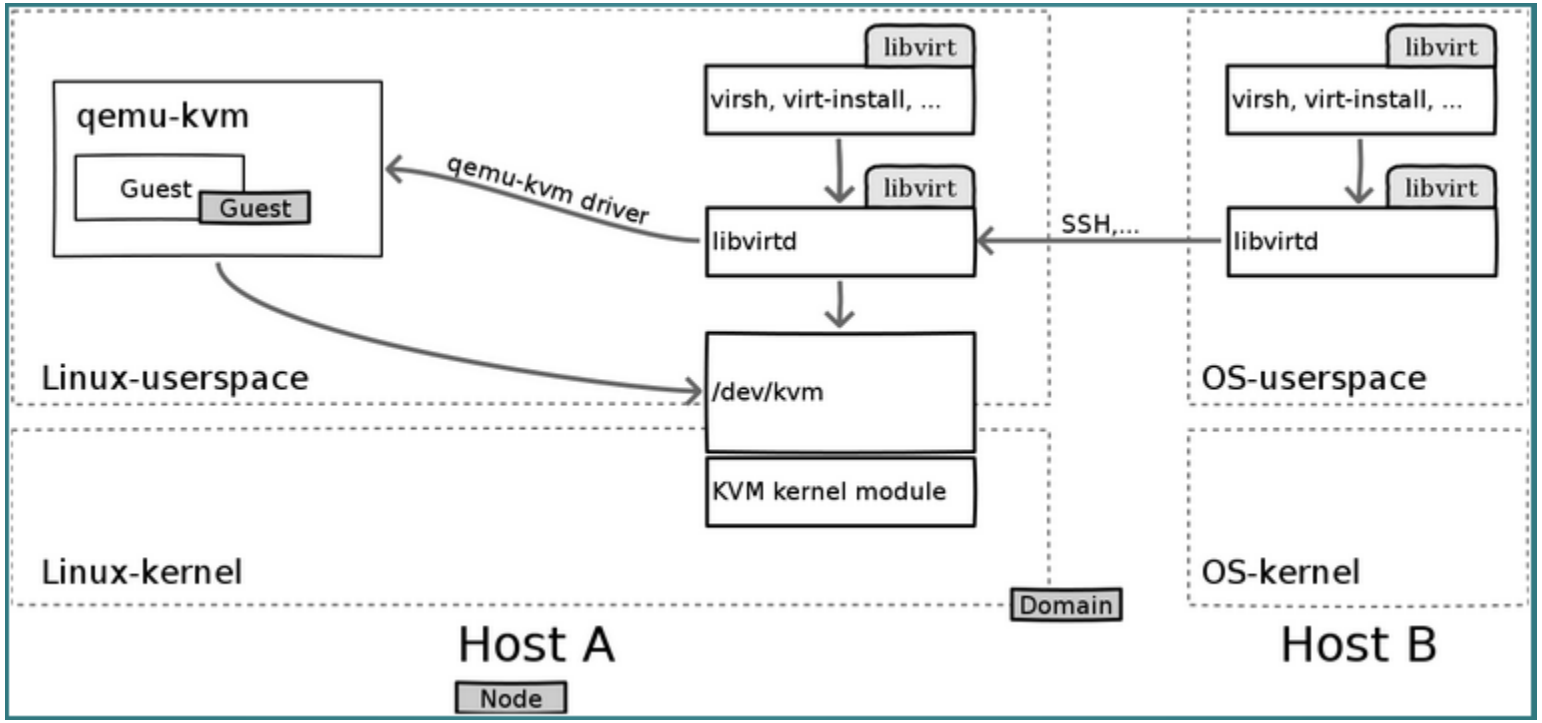
1.2 KVM Qemu Libvirt之间的关系

- libvirt是目前使用最为广泛的对KVM虚拟机进行管理的工具和API



1.2 KVM Qemu Libvirt之间的关系

- 总结：
  - Libvirtd是一个daemon进程，可以被本地的virsh调用，也可以被远程的virsh调用
  - Libvirtd调用qemu-kvm操作虚拟机
  - 有关CPU虚拟化的部分，qemu-kvm调用kvm的内核模块来实现



实验二：安装KVM, Qemu, Libvirt

- 查看内核模块中是否含有kvm, ubuntu默认加载这些模块

```
root@popsuper1982:/home/openstack# lsmod | grep kvm
kvm_intel             143060  0
kvm                   451511  1 kvm_intel
```

- 安装qemu-kvm

```
root@Nagios-Server:/home/openstack# apt-get install qemu-kvm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  acl cpu-checker ipxe-qemu libaio1 libasound2 libasound2-data libasyncns0 libbluetooth3 libboost-system1.54.0 libboost-thread1.54.0
  libbrlapi0.6 libcaca0 libfdt1 libflac8 libnspr4 libnss3 libnss3-nssdb libogg0 libpixman-1-0 libpulse0 librados2 librbd1 libsdl1.2debian
  libseccomp2 libsndfile1 libspice-server1 libusbredirparser1 libvorbis0a libvorbisenc2 libxen-4.4 libxenstore3.0 libyajl2 msr-tools
  qemu-keymaps qemu-system-common qemu-system-x86 qemu-utils seabios sharutils
```

```
root@Nagios-Server:/home/openstack# qemu-
qemu-img                  qemu-make-debian-root     qemu-system-i386          qemu-system-x86_64-spice
qemu-io                   qemu-nbd                  qemu-system-x86_64
```

```
root@Nagios-Server:/home/openstack# dpkg -l | grep qemu
ii  ipxe-qemu                         1.0.0+git-20131111.c3d1e78-2ubuntu1 all          PXE boot firmware - ROM images for qemu
ii  qemu-keymaps                      2.0.0+dfsg-2ubuntu1.3               all          QEMU keyboard maps
ii  qemu-kvm                          2.0.0+dfsg-2ubuntu1.3               amd64        QEMU Full virtualization on x86 hardware (transitional
package)
ii  qemu-system-common                2.0.0+dfsg-2ubuntu1.3               amd64        QEMU full system emulation binaries (common files)
ii  qemu-system-x86                   2.0.0+dfsg-2ubuntu1.3               amd64        QEMU full system emulation binaries (x86)
ii  qemu-utils                        2.0.0+dfsg-2ubuntu1.3               amd64        QEMU utilities
```

实验二：安装KVM, Qemu, Libvirt

- 安装libvirt

```
root@Nagios-Server:/home/openstack# apt-get install libvirt-bin
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  augeas-lenses bridge-utils cgroup-lite dnsmasq-base ebtables iproute libaugeas0 libmnl0 libnetcf1 libnetfilter-conntrack3 libnl-route-3-200
  libpciaccess0 libvirt0 libx86-1 libxml2-utils libxslt1.1 pm-utils vbetool
```

- 安装virt-install

```
root@Nagios-Server:/home/openstack# apt-get install virtinst
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  python-libvirt python-libxml2 python-pycurl python-urlgrabber
Suggested packages:
  libcurl4-gnutls-dev python-pycurl-dbg virt-viewer
The following NEW packages will be installed:
  python-libvirt python-libxml2 python-pycurl python-urlgrabber virtinst
```

Qemu-KVM

接下来的章节介绍qemu-kvm，先不介绍libvirt

2.1 QEMU-KVM: 安装第一个能上网的虚拟机

- 下载ubuntu的iso：http://www.ubuntu.com/download/cloud

  wget http://mirror.jmu.edu/pub/ubuntu-iso/14.04.1/ubuntu-14.04.1-server-amd64.iso

- 创建Image

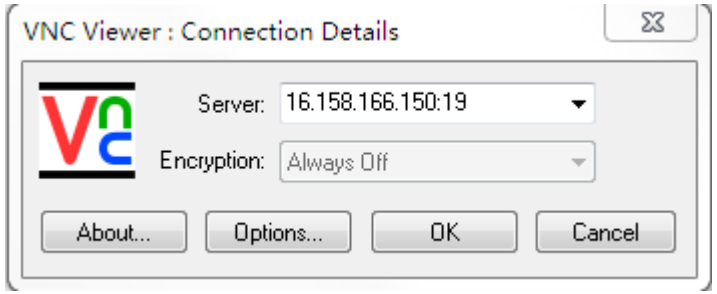
  qemu-img create -f qcow2 ubuntutest.img 5G

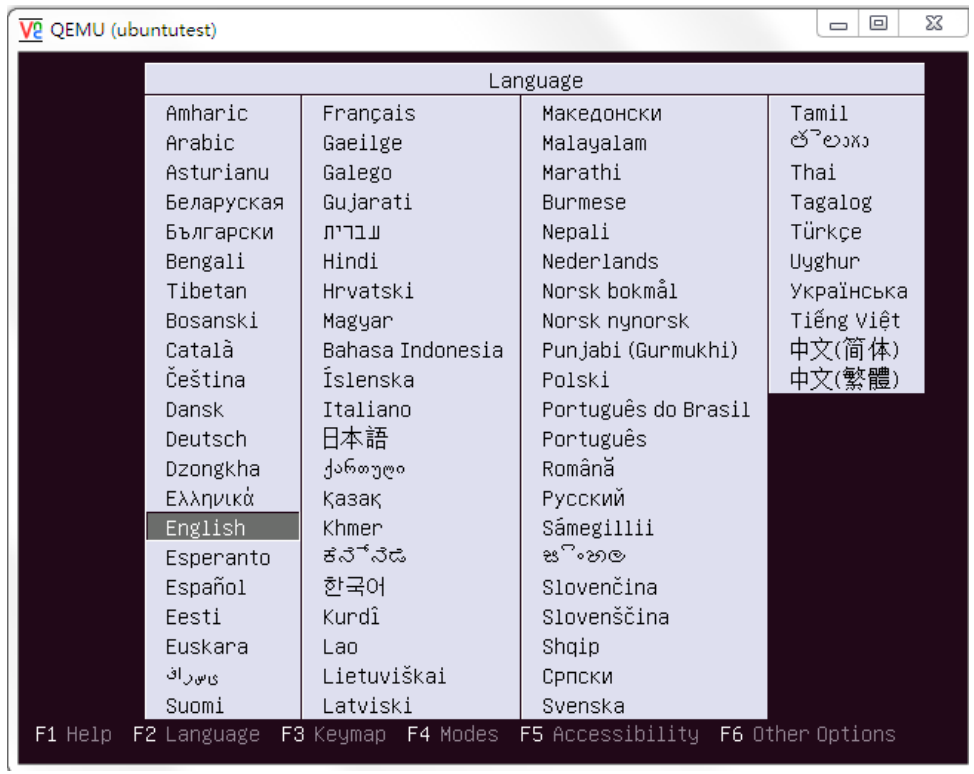
- 安装虚拟机

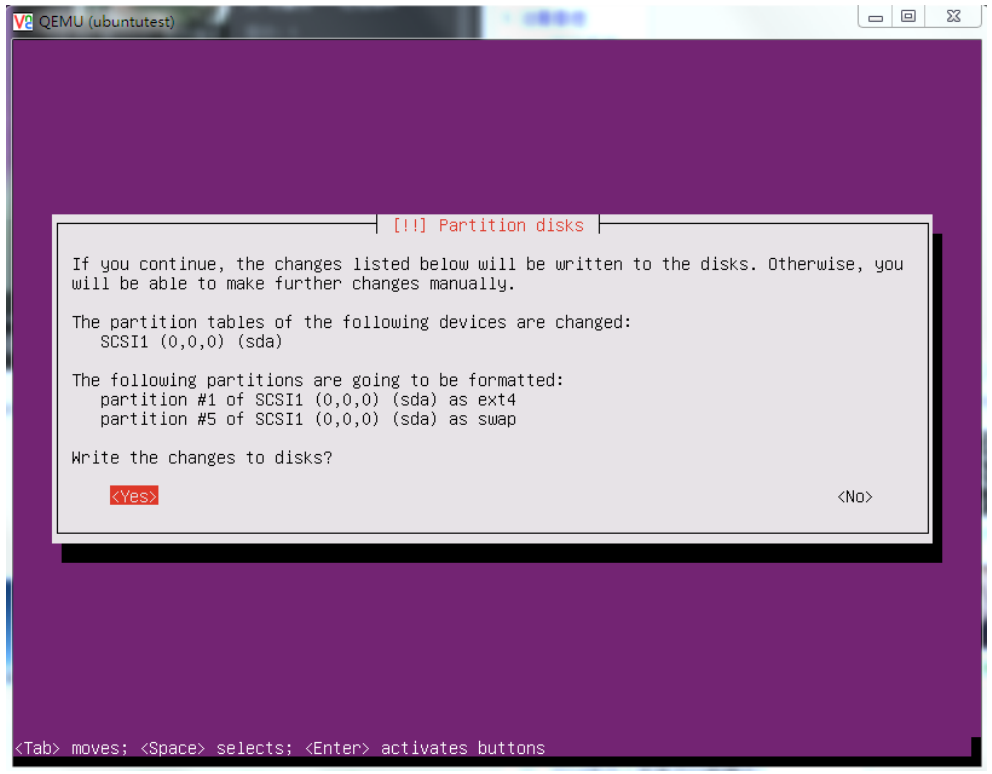
  qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.img -cdrom ubuntu-14.04-server-amd64.iso -boot d -vnc :19

2.1 QEMU-KVM:安装第一个能上网的虚拟机

- 连接到VNC，按照普通安装ubuntu的流程安装好ubuntu







2.1 QEMU-KVM: 安装第一个能上网的虚拟机

- 重新启动后，从第一个硬盘boot，然后关机shutdown -h now
- 压缩硬盘，我们后面会做很多实验，所以备份一份

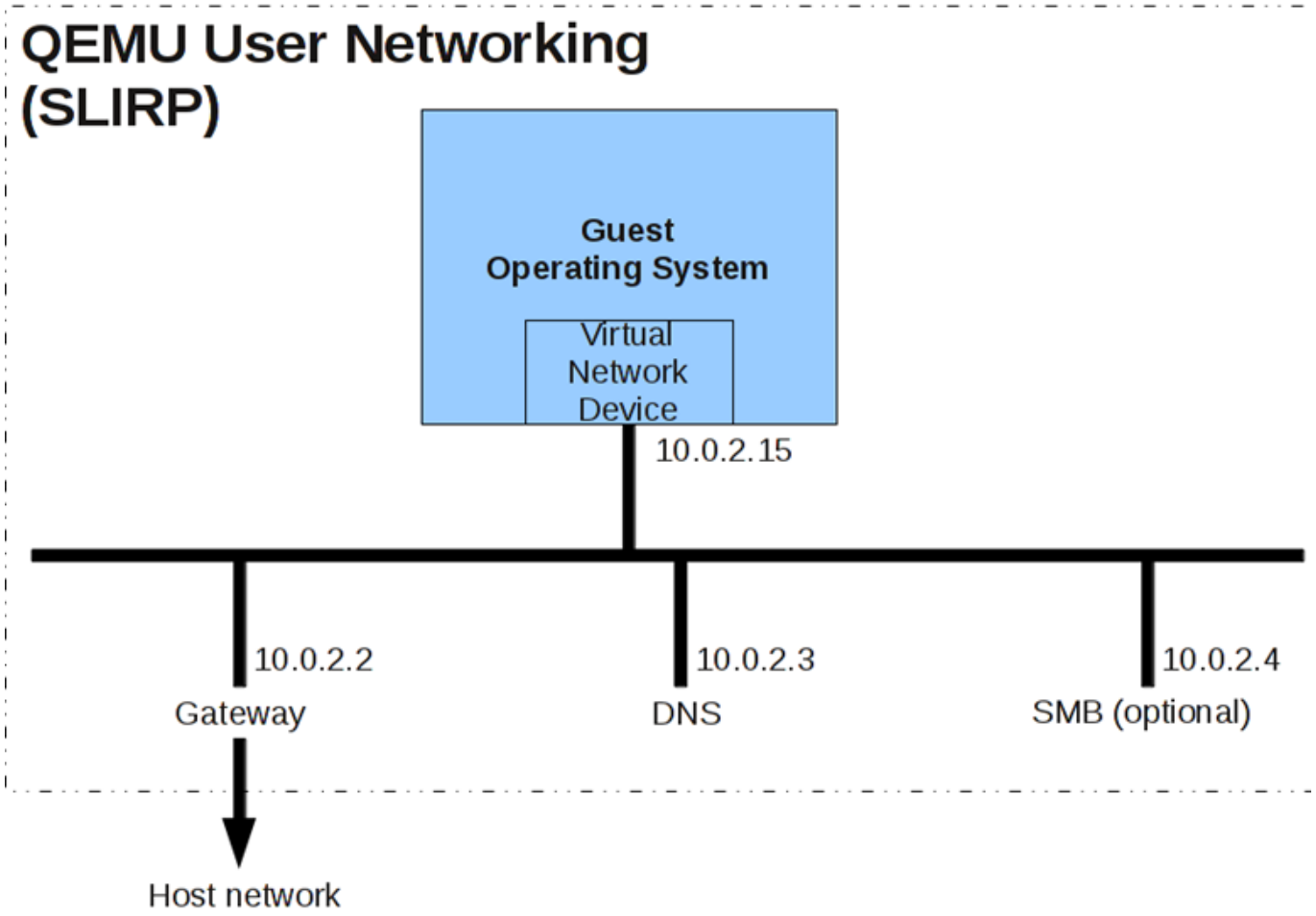
qemu-img convert -O qcow2 -c ubuntutest.img ubuntutest.qcow2

```
root@popsuper1982:/home/openstack/images# ls -lh ubuntutest.*
-rw-r--r-- 1 root root 1.4G Sep 21 07:28 ubuntutest.img
-rw-r--r-- 1 root root 453M Sep 21 07:30 ubuntutest.qcow2
```

- 从ubuntutest.qcow2启动虚拟机
- Qemu常用的网络配置
  - User Network
  - Tap + bridge network

2.1 QEMU-KVM: 安装第一个能上网的虚拟机

- User Network
  - 默认配置，简单易用
  - 性能较差
  - 不支持ICMP
  - 不能直接联网



2.1 QEMU-KVM:安装第一个能上网的虚拟机

- 配置Bridge Networking
  - 这个课程和openvswitch无关，所以不用它
  - 这里仅仅是简单的介绍，详细的后面会讲
  - (1) 在Host机器上创建bridge br0
    brctl addbr br0
  - (2) 将br0设为up
    ip link set br0 up
  - (3) 创建tap device
    tunctl -b
  - (4) 将tap0设为up
    ip link set tap0 up
  - (5) 将tap0加入到br0上
    brctl addif br0 tap0

2.1 QEMU-KVM:安装第一个能上网的虚拟机

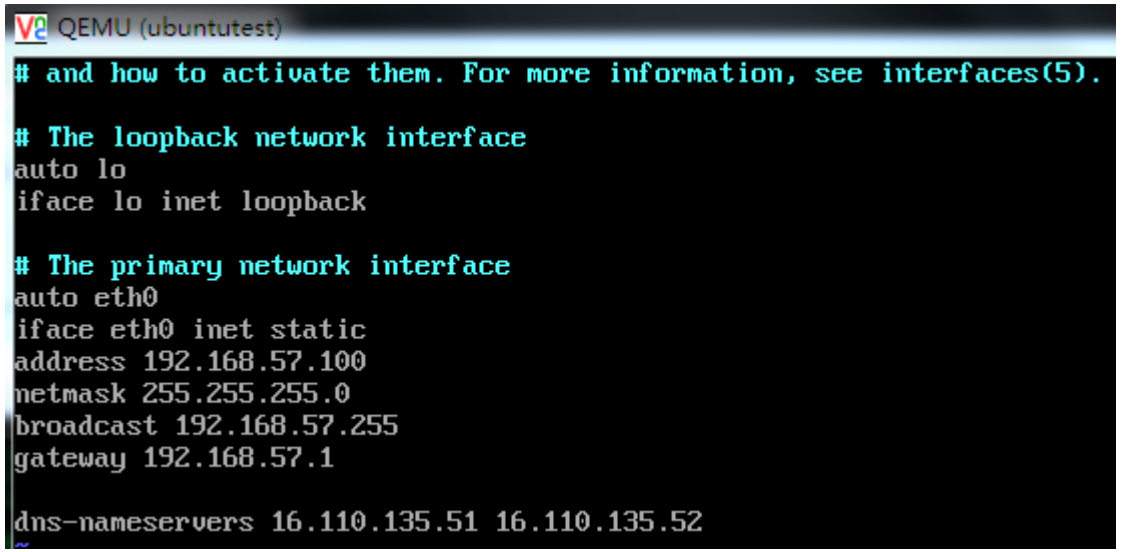
- (6) 启动虚拟机, 虚拟机连接tap0, tap0连接br0

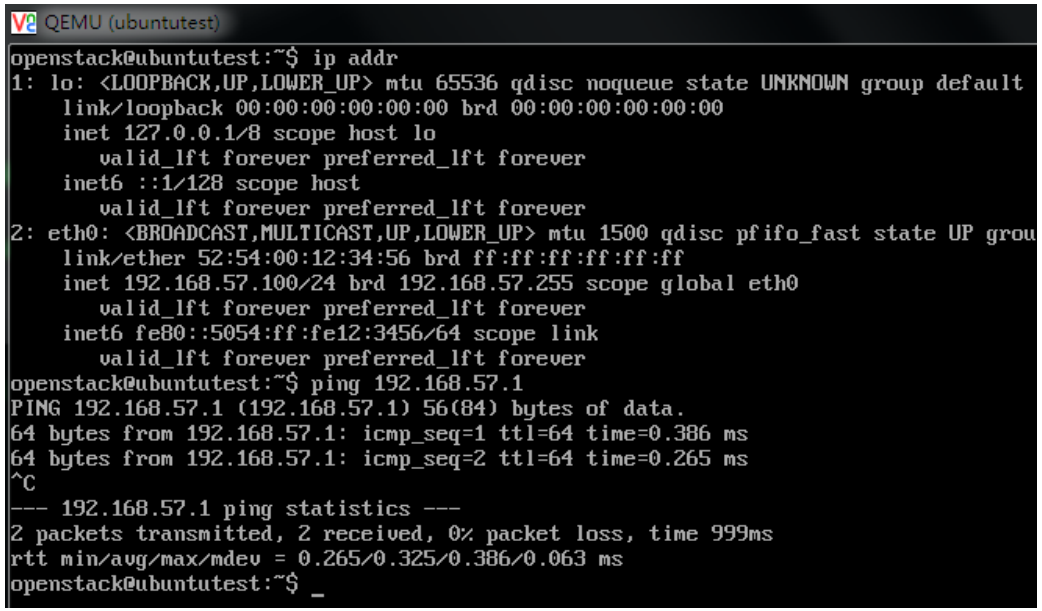
qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no

- (7) 虚拟机启动后，网卡没有配置，所以无法连接外网，先给br0设置一个ip

ifconfig br0 192.168.57.1/24

- (8) VNC连上虚拟机，给网卡设置地址，重启虚拟机，可ping通br0





2.1 QEMU-KVM:安装第一个能上网的虚拟机

- (9) 要想访问外网，在Host上设置NAT，并且enable ip forwarding，可以ping通外网网关

```
# sysctl -p
net.ipv4.ip_forward = 1

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

- (10) 如果DNS没配错，可以进行apt-get update

2.2 QEMU-KVM: 使用qemu monitor来管理虚拟机

- 访问qemu monitor console
  - 用VNC连接到虚拟机, Ctrl+Alt+2进入，Ctrl+Alt+1返回普通console
  - 在 QEMU 启动的时候指定 -monitor 参数。比如 -monitor stdio 将允许使用标准输入作为 monitor 命令源

qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio

- Help可以查看所有的命令
- 这里列一个总表，在以后对虚拟机的管理中分别要用到
- ACL
  - acl_add aclname match allow|deny [index]
  - acl_policy aclname allow|deny
  - acl_remove aclname match
  - acl_reset aclname
  - acl_show aclname

2.2 QEMU-KVM: 使用qemu monitor来管理虚拟机

- Memory
  - balloon target: request VM to change its memory allocation
  - dump-guest-memory [-p] filename [begin] [length]
  - memsave addr size file: save to disk virtual memory dump
  - pmemsave addr size file: save to disk physical memory dump
  - sum addr size: compute the checksum of a memory region
  - x /fmt addr: virtual memory dump starting at 'addr'
  - xp /fmt addr: physical memory dump starting at 'addr‘
- CPU
  - cpu index -- set the default CPU
  - cpu-add id -- add cpu
- Device
  - device_add driver[,prop=value][,...]
  - device_del device
  - change device filename [format]
  - eject [-f] device
  - boot_set bootdevice
  - commit device|all

2.2 QEMU-KVM: 使用qemu monitor来管理虚拟机

- Drive
  - drive_add
  - drive_backup [-n] [-f] device target [format]
  - drive_del device
  - drive_mirror [-n] [-f] device target [format]
- Block Device
  - block_passwd block_passwd device password
  - block_resize device size
  - block_set_io_throttle device bps bps_rd bps_wr iops iops_rd iops_wr
  - block_stream device [speed [base]]
  - block_job_cancel [-f] device
  - block_job_complete device
  - block_job_pause device
  - block_job_resume device
  - block_job_cancel [-f] device
- Drive和Block的区别是Drive是站在HOST的角度来看的，Block Device是站在虚拟机的角度去看的

2.2 QEMU-KVM: 使用qemu monitor来管理虚拟机

- Character Device
  - chardev-add args -- add chardev
  - chardev-remove id -- remove chardev
- VNC
  - set_password protocol password: set spice/vnc password
  - expire_password protocol time: set spice/vnc password expire-time
- Mouse
  - mouse_button state: change mouse button state (1=L, 2=M, 4=R)
  - mouse_move dx dy [dz] : send mouse move events
  - mouse_set index : set which mouse device receives events

2.2 QEMU-KVM: 使用qemu monitor来管理虚拟机

- Snapshot
  - savevm [tag|id] -- save a VM snapshot.
  - loadvm tag|id -- restore a VM snapshot
  - delvm tag|id -- delete a VM snapshot
  - snapshot_blkdev_internal device name -- take an internal snapshot of device.
  - snapshot_delete_blkdev_internal device name [id] -- delete an internal snapshot of device.
  - snapshot_blkdev [-n] device [new-image-file] [format] -- initiates a live snapshot of device. If a new image file is specified, the new image file will become the new root image.
- Network Block Device
  - nbd_server_add nbd_server_add [-w] device -- export a block device via NBD
  - nbd_server_start nbd_server_start [-a] [-w] host:port -- serve block devices on the given host and port
  - nbd_server_stop nbd_server_stop -- stop serving block devices using the NBD protocol

2.2 QEMU-KVM: 使用qemu monitor来管理虚拟机

- Network
  - host_net_add tap|user|socket|vde|netmap|dump [options] -- add host VLAN client
  - host_net_remove vlan_id name -- remove host VLAN client
  - hostfwd_add [vlan_id name] [tcp|udp]:[hostaddr]:hostport-[guestaddr]:guestport -- redirect TCP or UDP connections from host to guest (requires -net user)
  - hostfwd_remove [vlan_id name] [tcp|udp]:[hostaddr]:hostport -- remove host-to-guest TCP or UDP redirection
  - netdev_add [user|tap|socket|hubport|netmap],id=str[,prop=value][,...] -- add host network device
  - netdev_del id -- remove host network device
  - set_link name on|off -- change the link status of a network adapter
- USB
  - usb_add device -- add USB device (e.g. 'host:bus.addr' or 'host:vendor_id:product_id')
  - usb_del device -- remove USB device 'bus.addr'

2.2 QEMU-KVM: 使用qemu monitor来管理虚拟机

- Migrate
  - migrate [-d] [-b] [-i] uri -- migrate to URI
  - migrate_cancel -- cancel the current VM migration
  - migrate_set_cache_size value -- set cache size (in bytes)
  - migrate_set_capability capability state -- Enable/Disable the usage of a capability for migration
  - migrate_set_downtime value -- set maximum tolerated downtime (in seconds) for migrations
  - migrate_set_speed value -- set maximum speed (in bytes) for migrations.
- System
  - system_powerdown -- send system power down event
  - system_reset -- reset the system
  - system_wakeup -- wakeup guest from suspend

实验三：使用qemu monitor查看信息

(qemu) info
info balloon -- show balloon information
info block [-v] [device] -- show info of one block device or all block devices (and details of images with -v option)
info block-jobs -- show progress of ongoing block device operations
info blockstats -- show block device statistics
info capture -- show capture information
info chardev -- show the character devices
info cpus -- show infos for each CPU
info cpustats -- show CPU statistics
info history -- show the command line history
info irq -- show the interrupts statistics (if available)
info jit -- show dynamic compiler info
info kvm -- show KVM information
info mem -- show the active virtual memory mappings
info mice -- show which guest mouse is receiving events
info migrate -- show migration status
info migrate_cache_size -- show current migration xbzrle cache size
info migrate_capabilities -- show current migration capabilities
info mtree -- show memory tree
info name -- show the current VM name
info network -- show the network state
info numa -- show NUMA information

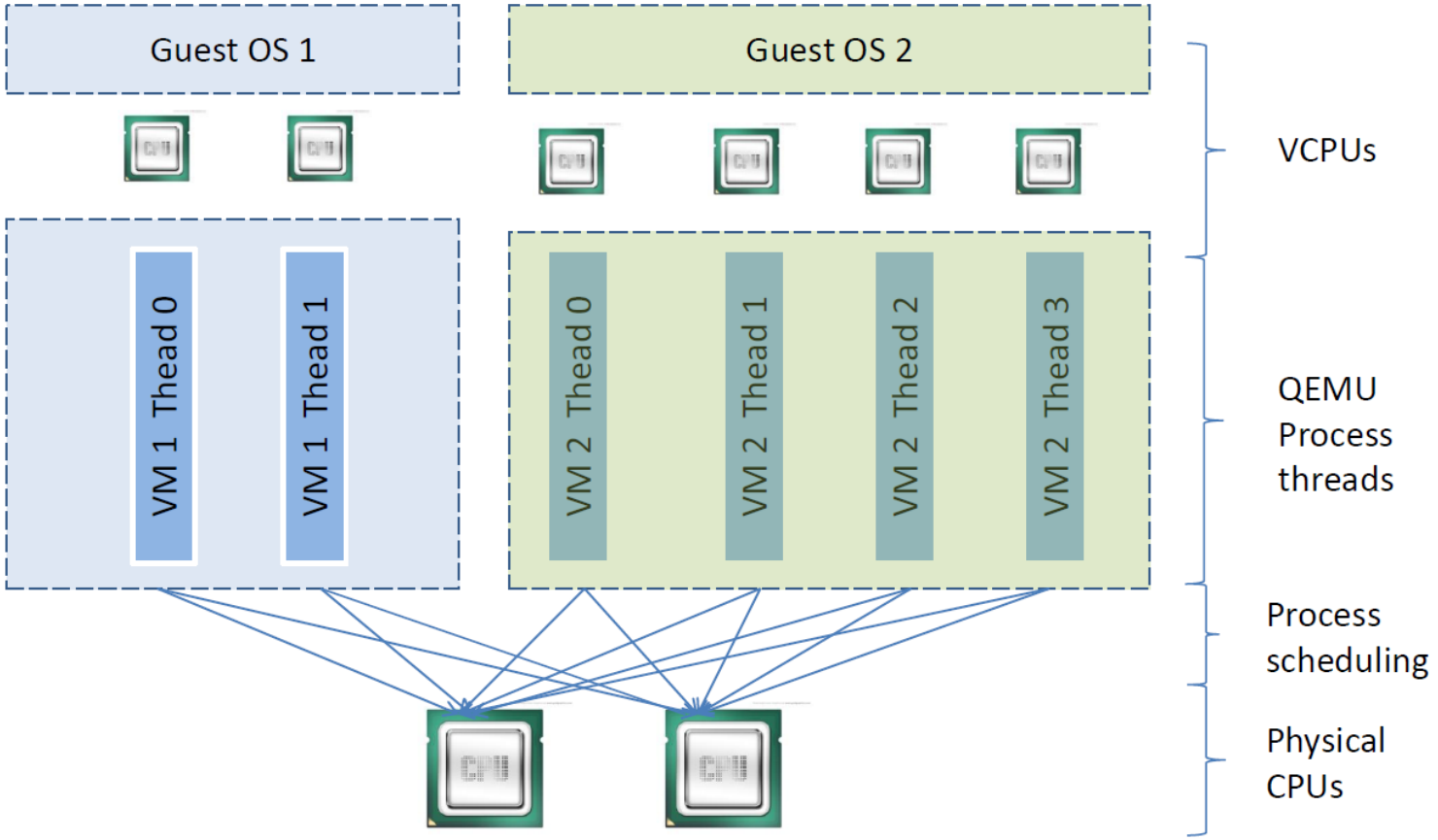
实验三：使用qemu monitor查看信息

info pci -- show PCI info
info pcmcia -- show guest PCMCIA status
info pic -- show i8259 (PIC) state
info profile -- show profiling information
info qdm -- show qdev device model list
info qtree -- show device tree
info registers -- show the cpu registers
info roms -- show roms
info snapshots -- show the currently saved VM snapshots
info spice -- show the spice server status
info status -- show the current VM status (running|paused)
info tlb -- show virtual to physical memory mappings
info tpm -- show the TPM device
info trace-events -- show available trace-events & their state
info usb -- show guest USB devices
info usbhost -- show host USB devices
info usernet -- show user network stack connection states
info uuid -- show the current VM UUID
info version -- show the version of QEMU
info vnc -- show the vnc server status

实验三：使用qemu monitor查看信息

- 查看CPU: 每个CPU对应于HOST上的一个线程

```
(qemu) info cpus
* CPU #0: pc=0xffffffff8104f596 (halted) thread_id=22068
```



实验三：使用qemu monitor查看信息

- 查看Block设备: 看到Image文件

```
(qemu) info block
ide0-hd0: ubuntutest.qcow2 (qcow2)

ide1-cd0: [not inserted]
    Removable device: not locked, tray closed

floppy0: [not inserted]
    Removable device: not locked, tray closed

sd0: [not inserted]
    Removable device: not locked, tray closed
```

- 查看Block Status

```
(qemu) info blockstats
ide0-hd0: rd_bytes=141093376 wr_bytes=1306624 rd_operations=57037 wr_operations=138 flush_operations=27 wr_total_time_ns=326844399 rd_total_time_
ns=2871860075 flush_total_time_ns=551998560
```

- 查看Network

```
(qemu) info network
hub 0
 \ tap.0: index=0,type=tap,ifname=tap0,script=no,downscript=no
 \ virtio-net-pci.0: index=0,type=nic,model=virtio-net-pci,macaddr=52:54:00:12:34:56
```

实验三：使用qemu monitor查看信息

- 查看所有的Device
  - info qtree

2.3 QEMU-KVM: qemu的硬件虚拟化

- Qemu可以模拟物理计算机常用的硬件设备
  - 计算机体系结构
  - CPU
  - SMP对称多处理器
  - System Management BIOS
  - 内存
  - System Clock
  - USB
  - 设备总线
  - 显示器
  - 声卡
  - 网卡
  - CD-ROM
  - 硬盘

Openstack虚拟机的参数

```
qemu-system-x86_64
-enable-kvm
-name instance-00000024
-machine pc-i440fx-trusty,accel=kvm,usb=off
-cpu SandyBridge,+erms,+smep,+fsgsbase,+pdpe1gb,+rdrand,+f16c,+osxsave,+dca,+pcid,+pdcm,+xtpr,+tm2,+est,+smx,+vmx,+ds_cpl,+monitor,+dtes64,+pbe,+tm,+ht,+ss,+acpi,+ds,+vme
-m 2048 -realtime mlock=off
-smp 1,sockets=1,cores=1,threads=1
-uuid 1f8e6f7e-5a70-4780-89c1-464dc0e7f308
-smbios type=1,manufacturer=OpenStack Foundation,product=OpenStack Nova,version=2014.1,serial=80590690-87d2-e311-b1b0-a0481cabdfb4,uuid=1f8e6f7e-5a70-4780-89c1-
464dc0e7f308
-no-user-config
-nodefaults
-chardev socket,id=charmonitor,path=/var/lib/libvirt/qemu/instance-00000024.monitor,server,nowait
-mon chardev=charmonitor,id=monitor,mode=control
-rtc base=utc,driftfix=slew
-global kvm-pit.lost_tick_policy=discard
-no-hpet
-no-shutdown
-boot strict=on
-device piix3-usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2
-drive file=/var/lib/nova/instances/1f8e6f7e-5a70-4780-89c1-464dc0e7f308/disk,if=none,id=drive-virtio-disk0,format=qcow2,cache=none
-device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-virtio-disk0,id=virtio-disk0,bootindex=1
-netdev tap,fd=32,id=hostnet0,vhost=on,vhostfd=37
-device virtio-net-pci,netdev=hostnet0,id=net0,mac=fa:16:3e:d1:2d:99,bus=pci.0,addr=0x3
-chardev file,id=charserial0,path=/var/lib/nova/instances/1f8e6f7e-5a70-4780-89c1-464dc0e7f308/console.log
-device isa-serial,chardev=charserial0,id=serial0
-chardev pty,id=charserial1
-device isa-serial,chardev=charserial1,id=serial1
-device usb-tablet,id=input0
-vnc 0.0.0.0:12
-k en-us
-device cirrus-vga,id=video0,bus=pci.0,addr=0x2
-device virtio-balloon-pci,id=balloon0,bus=pci.0,addr=0x5
```

2.3 QEMU-KVM: qemu的硬件虚拟化

- 计算机体系结构
  - QEMU会模拟多种Processor architectures，常用的有：
    - PC (x86 or x86_64 processor)
    - Mac99 PowerMac (PowerPC processor)
    - Sun4u/Sun4v (64-bit Sparc processor)
    - MIPS magnum (64-bit MIPS processor)
  - qemu-system-x86_64 --machine ?

所以在命令行中有
-machine pc-i440fx-trusty,accel=kvm,usb=off

如果使用KVM hardware-assisted virtualization，也即BIOS中VD-T是打开的，则参数中accel=kvm

如果不使用hardware-assisted virtualization，用的是纯模拟，则有参数accel = tcg，-no-kvm

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 --machine ?
Supported machines are:
pc-0.13              Standard PC (i440FX + PIIX, 1996)
pc-i440fx-2.0        Standard PC (i440FX + PIIX, 1996)
pc-1.0               Standard PC (i440FX + PIIX, 1996)
pc-q35-1.7           Standard PC (Q35 + ICH9, 2009)
pc-1.1               Standard PC (i440FX + PIIX, 1996)
q35                  Standard PC (Q35 + ICH9, 2009) (alias of pc-q35-2.0)
pc-q35-2.0           Standard PC (Q35 + ICH9, 2009)
pc-i440fx-1.4        Standard PC (i440FX + PIIX, 1996)
pc-i440fx-1.5        Standard PC (i440FX + PIIX, 1996)
pc-0.14              Standard PC (i440FX + PIIX, 1996)
pc-0.15              Standard PC (i440FX + PIIX, 1996)
xenfv                Xen Fully-virtualized PC
pc-q35-1.4           Standard PC (Q35 + ICH9, 2009)
isapc                ISA-only PC
pc-0.10              Standard PC (i440FX + PIIX, 1996)
pc                   Ubuntu 14.04 PC (i440FX + PIIX, 1996) (alias of pc-i440fx-trusty)
pc-i440fx-trusty     Ubuntu 14.04 PC (i440FX + PIIX, 1996) (default)
pc-1.2               Standard PC (i440FX + PIIX, 1996)
pc-0.11              Standard PC (i440FX + PIIX, 1996)
pc-i440fx-1.7        Standard PC (i440FX + PIIX, 1996)
pc-i440fx-1.6        Standard PC (i440FX + PIIX, 1996)
none                 empty machine
xenpv                Xen Para-virtualized PC
pc-q35-1.5           Standard PC (Q35 + ICH9, 2009)
pc-q35-1.6           Standard PC (Q35 + ICH9, 2009)
pc-0.12              Standard PC (i440FX + PIIX, 1996)
pc-1.3               Standard PC (i440FX + PIIX, 1996)
```

2.3 QEMU-KVM: qemu的硬件虚拟化

- CPU
  - 下面的命令可以列出支持的CPU： qemu-system-x86_64 --cpu ?

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 --cpu ?
x86           qemu64  QEMU Virtual CPU version 2.0.0
x86           phenom  AMD Phenom(tm) 9550 Quad-Core Processor
x86         core2duo  Intel(R) Core(TM)2 Duo CPU     T7700  @ 2.40GHz
x86            kvm64  Common KVM processor
x86           qemu32  QEMU Virtual CPU version 2.0.0
x86            kvm32  Common 32-bit KVM processor
x86          coreduo  Genuine Intel(R) CPU           T2600  @ 2.16GHz
x86              486
x86          pentium
x86         pentium2
x86         pentium3
x86           athlon  QEMU Virtual CPU version 2.0.0
x86             n270  Intel(R) Atom(TM) CPU N270   @ 1.60GHz
x86           Conroe  Intel Celeron_4x0 (Conroe/Merom Class Core 2)
x86           Penryn  Intel Core 2 Duo P9xxx (Penryn Class Core 2)
x86          Nehalem  Intel Core i7 9xx (Nehalem Class Core i7)
x86         Westmere  Westmere E56xx/L56xx/X56xx (Nehalem-C)
x86      SandyBridge  Intel Xeon E312xx (Sandy Bridge)
x86          Haswell  Intel Core Processor (Haswell)
x86       Opteron_G1  AMD Opteron 240 (Gen 1 Class Opteron)
x86       Opteron_G2  AMD Opteron 22xx (Gen 2 Class Opteron)
x86       Opteron_G3  AMD Opteron 23xx (Gen 3 Class Opteron)
x86       Opteron_G4  AMD Opteron 62xx class CPU
x86       Opteron_G5  AMD Opteron 63xx class CPU
x86             host  KVM processor with all supported host features (only available in KVM mode)

Recognized CPUID flags:
  pbe ia64 tm ht ss sse2 sse fxsr mmx acpi ds clflush pn pse36 pat cmov mca pge mtrr sep apic cx8 mce pae msr tsc pse de vme fpu
  hypervisor rdrand f16c avx osxsave xsave aes tsc-deadline popcnt movbe x2apic sse4.2|sse4_2 sse4.1|sse4_1 dca pcid pdcm xtpr cx16 fma cid ssse3
 tm2 est smx vmx ds_cpl monitor dtes64 pclmulqdq|pclmuldq pni|sse3
  smap adx rdseed rtm invpcid erms bmi2 smep avx2 hle bmi1 fsgsbase
  3dnow 3dnowext lm|i64 rdtscp pdpe1gb fxsr_opt|ffxsr mmxext nx|xd syscall
  perfctr_nb perfctr_core topoext tbm nodeid_msr tce fma4 lwp wdt skinit xop ibs osvw 3dnowprefetch misalignsse sse4a abm cr8legacy extapic svm c
mp_legacy lahf_lm
  pmm-en pmm phe-en phe ace2-en ace2 xcrypt-en xcrypt xstore-en xstore
  kvm_pv_unhalt kvm_pv_eoi kvm_steal_time kvm_asyncpf kvmclock kvm_mmu kvm_nopiodelay kvmclock
  pfthreshold pause_filter decodeassists flushbyasid vmcb_clean tsc_scale nrip_save svm_lock lbrv npt
```

所以参数中有下面的CPU设置及添加的flag

-cpu SandyBridge,+erms,+smep,+fsgsbase,+pdpe1gb,+rdrand,+f16c,+osxsave,+dca,+pcid,+pdcm,+xtpr,+tm2,+est,+smx,+vmx,+ds_cpl,+monitor,+dtes64,+pbe,+tm,+ht,+ss,+acpi,+ds,+vme

2.3 QEMU-KVM: qemu的硬件虚拟化

- SMP对称多处理器
  - 对称多处理"（Symmetrical Multi-Processing）简称SMP，是指在一个计算机上汇集了一组处理器(多CPU),各CPU之间共享内存子系统以及总线结构。
  - QEMU and KVM 可以最多模拟SMP到255个CPU.
  - 我们的参数中有-smp 1,sockets=1,cores=1,threads=1
    - qemu仿真了一个具有1个vcpu，一个socket，一个core，一个threads的处理器。
  - socket, core, threads是什么概念呢
    - socket就是主板上插cpu的槽的数目，也即常说的“路”
    - core就是我们平时说的“核”，即双核，4核等
    - thread就是每个core的硬件线程数，即超线程
  - 具体例子，某个服务器是：2路4核超线程(一般默认为2个线程)，通过cat /proc/cpuinfo看到的是2*4*2=16个processor，很多人也习惯成为16核了！

2.3 QEMU-KVM: qemu的硬件虚拟化

- SMBIOS
  - SMBIOS全称System Management BIOS，用于表示x86 architectures的硬件信息
  - 在unix系统上，可以使用命令dmidecode得到，SMBIOS的信息被分为多个table中，称为type
  - type 0是BIOS信息

```
root@popsuper1982:/home/openstack/images# dmidecode --type 0
# dmidecode 2.12
SMBIOS 2.6 present.

Handle 0x0009, DMI type 0, 24 bytes
BIOS Information
        Vendor: Hewlett-Packard
        Version: 68CCU Ver. F.12
        Release Date: 03/09/2011
        Address: 0xF0000
        Runtime Size: 64 kB
        ROM Size: 3072 kB
        Characteristics:
                PCI is supported
                PC Card (PCMCIA) is supported
                BIOS is upgradeable
                BIOS shadowing is allowed
                Boot from CD is supported
                Selectable boot is supported
                EDD is supported
                Print screen service is supported (int 5h)
                8042 keyboard services are supported (int 9h)
                Serial services are supported (int 14h)
                Printer services are supported (int 17h)
                ACPI is supported
                USB legacy is supported
                Smart battery is supported
                BIOS boot specification is supported
                Function key-initiated network boot is supported
                Targeted content distribution is supported
        BIOS Revision: 15.18
        Firmware Revision: 48.49
```

2.3 QEMU-KVM: qemu的硬件虚拟化

type 1是系统信息

```
root@popsuper1982:/home/openstack/images# dmidecode --type 1
# dmidecode 2.12
SMBIOS 2.6 present.

Handle 0x000A, DMI type 1, 27 bytes
System Information
        Manufacturer: Hewlett-Packard
        Product Name: HP EliteBook 8440p
        Version:
        Serial Number: CND0480RQQ
        UUID: F27F4A00-ECFA-11DE-81D0-B9F76E39343C
        Wake-up Type: Power Switch
        SKU Number: SJ017UC#B1L
        Family: 103C_5336AN
```

type 2是主板信息

```
root@popsuper1982:/home/openstack/images# dmidecode --type 2
# dmidecode 2.12
SMBIOS 2.6 present.

Handle 0x000B, DMI type 2, 16 bytes
Base Board Information
        Manufacturer: Hewlett-Packard
        Product Name: 172A
        Version: KBC Version 30.31
        Serial Number: CND0480RQQ
        Asset Tag: Not Specified
        Features:
                Board is a hosting board
                Board is replaceable
        Location In Chassis:
        Chassis Handle: 0x000C
        Type: Unknown
        Contained Object Handles: 0
```

所以有参数

-smbios type=1,manufacturer=OpenStack Foundation,product=OpenStack Nova,version=2014.1,serial=80590690-87d2-e311-b1b0-a0481cabdfb4,uuid=1f8e6f7e-5a70-4780-89c1-464dc0e7f308

指定了type 1的信息

2.3 QEMU-KVM: qemu的硬件虚拟化

- RAM
  - 内存的大小在参数中，用-m来指定
  - -m 2048
  - guest真正运行态的占用的内存的大小，是用Memory Ballooning来调整
  - -device virtio-balloon-pci,id=balloon0,bus=pci.0,addr=0x5
- ACPI (Advanced Configuration and Power Interface)
  - open industry standard for power management
  - ACPI disabled with the option -no-acpi
- System Clock
  - 系统时间由参数-rtc指定 -rtc [base = utc | localtime | date] [, clock = host | vm] [, driftfix = none | slew]

    -rtc base=utc,driftfix=slew
  - High Precision Event Timer (HPET)是可以更准确的设定时间的

    -no-hpet

2.3 QEMU-KVM: qemu的硬件虚拟化

- USB
  - USB的好处就是即插即用
  - 参数-usb，可以模拟一个PCI UHCI USB控制器
  - 参数里面看到的是
    - -device usb-tablet,id=input0
    - 通过tablet，鼠标可以在HOST和GUEST机器之间自由的切换
- 显示器
  - 用参数-vga设置，默认为cirrus，它模拟了CL-GD5446 PCI VGA card
  - -device cirrus-vga,id=video0,bus=pci.0,addr=0x2
- 声卡

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 -soundhw ?
Valid sound card names (comma separated):
ac97         Intel 82801AA AC97 Audio
adlib        Yamaha YM3812 (OPL2)
cs4231a      CS4231A
es1370       ENSONIQ AudioPCI ES1370
gus          Gravis Ultrasound GF1
hda          Intel HD Audio
pcspk        PC speaker
sb16         Creative Sound Blaster 16

-soundhw all will enable all of the above
```

2.3 QEMU-KVM: qemu的硬件虚拟化

- 网卡
  - 使用-net参数和-device
  - 从HOST角度：-netdev tap,fd=32,id=**hostnet0**,vhost=on,vhostfd=37
  - 从GUEST角度：-device virtio-net-pci,netdev=**hostnet0**,id=net0,mac=fa:16:3e:d1:2d:99,bus=pci.0,addr=0x3
- 硬盘
  - 使用-hda -hdb
  - 使用-drive和-device
  - 从HOST角度：-drive file=/var/lib/nova/instances/1f8e6f7e-5a70-4780-89c1-464dc0e7f308/disk,if=none,id=**drive-virtio-disk0**,format=qcow2,cache=none
  - 从GUEST角度：-device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x4,drive=**drive-virtio-disk0**,id=virtio-disk0,bootindex=1
- CDROM
  - 使用-cdrom
  - 如果想直接挂载Host机器上的CD-ROM -cdrom /dev/cdrom

2.3 QEMU-KVM: qemu的硬件虚拟化

- Character Device: console的log
  - 从HOST角度：-chardev file,id=**charserial0**,path=/var/lib/nova/instances/1f8e6f7e-5a70-4780-89c1-464dc0e7f308/console.log
  - 从GUEST角度：-device isa-serial,chardev=**charserial0**,id=serial0
  - 当然需要在image里面做如下的配置才能将boot log写入console
  - /boot/grub/grub.cfg中有
    - linux /boot/vmlinuz-3.2.0-49-virtual root=UUID=6d2231e4-0975-4f35-a94f-56738c1a8150 ro console=ttyS0
- Character Device: PTY
  - -chardev pty,id=charserial1
  - -device isa-serial,chardev=charserial1,id=serial1
- VNC
  - -vnc 0.0.0.0:12

2.3 QEMU-KVM: qemu的硬件虚拟化

- PCI(Peripheral Component Interconnect)是设备总线标准
  - 在参数中常看到bus和addr，是PCI总线的概念
  - 一个PCI子系统包含四个模块
    - Bus
      - 最多256个Bus，常用的是Bus 0和Bus 1
    - Device
      - Device就是连接到Bus上的设备，如声卡，网卡等，最多32个
    - Function
      - 每个Device有至少有一个Function 0，最多8个
    - Register
      - 每个Function有256 eight-bit registers
  - -device piix3-usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2表示：
    - 连接到Bus的pci.0
    - 是这个Bus的第一个Device, 0x1
    - 是这个Device的第二个Function, 0x2，如果只有一个Function，这个数字可以忽略

2.3 QEMU-KVM: qemu的硬件虚拟化

- 在monitor中info pci

```
Bus  0, device   0, function 0:
  Host bridge: PCI device 8086:1237
    id ""
Bus  0, device   1, function 0:
  ISA bridge: PCI device 8086:7000
    id ""
Bus  0, device   1, function 1:
  IDE controller: PCI device 8086:7010
    BAR4: I/O at 0xc0a0 [0xc0af].
    id ""
Bus  0, device   1, function 2:
  USB controller: PCI device 8086:7020
    IRQ 11.
    BAR4: I/O at 0xc040 [0xc05f].
    id "usb"
Bus  0, device   1, function 3:
  Bridge: PCI device 8086:7113
    IRQ 9.
    id ""
```

USB

-device piix3-usb-uhci,
id=usb,bus=pci.0,addr=0x1.0x2

```
Bus  0, device   2, function 0:
  VGA controller: PCI device 1013:00b8
    BAR0: 32 bit prefetchable memory at 0xfc000000 [0xfdffffff].
    BAR1: 32 bit memory at 0xfebd0000 [0xfebd0fff].
    BAR6: 32 bit memory at 0xffffffffffffffff [0x0000fffe].
    id "video0"
```

显示器

-device cirrus-vga,id=video0,bus=pci.0,addr=0x2

```
Bus  0, device   3, function 0:
  Ethernet controller: PCI device 1af4:1000
    IRQ 11.
    BAR0: I/O at 0xc060 [0xc07f].
    BAR1: 32 bit memory at 0xfebd1000 [0xfebd1fff].
    BAR6: 32 bit memory at 0xffffffffffffffff [0x0003fffe].
    id "net0"
```

网卡

-device virtio-net-pci,netdev=hostnet0,id=net0,
mac=fa:16:3e:d1:2d:99,bus=pci.0,addr=0x3

```
Bus  0, device   4, function 0:
  SCSI controller: PCI device 1af4:1001
    IRQ 11.
    BAR0: I/O at 0xc000 [0xc03f].
    BAR1: 32 bit memory at 0xfebd2000 [0xfebd2fff].
    id "virtio-disk0"
```

硬盘

-device virtio-blk-pci,scsi=off,bus=pci.0,
addr=0x4,drive=drive-virtio-disk0,
id=virtio-disk0,bootindex=1

```
Bus  0, device   5, function 0:
  Class 0255: PCI device 1af4:1002
    IRQ 10.
    BAR0: I/O at 0xc080 [0xc09f].
    id "balloon0"
```

内存

-device virtio-balloon-pci,id=balloon0,
bus=pci.0,addr=0x5

实验四：查看qemu所能模拟的设备

- qemu-system-x86_64 -device ?

```
Controller/Bridge/Hub devices:
name "i82801b11-bridge", bus PCI
name "ioh3420", bus PCI, desc "Intel IOH device id 3420 PCIE Root Port"
name "pci-bridge", bus PCI, desc "Standard PCI Bridge"
name "usb-host", bus usb-bus
name "usb-hub", bus usb-bus
name "x3130-upstream", bus PCI, desc "TI X3130 Upstream Port of PCI Express Switch"
name "xio3130-downstream", bus PCI, desc "TI X3130 Downstream Port of PCI Express Switch"
```

```
USB devices:
name "ich9-usb-ehci1", bus PCI
name "ich9-usb-ehci2", bus PCI
name "ich9-usb-uhci1", bus PCI
name "ich9-usb-uhci2", bus PCI
name "ich9-usb-uhci3", bus PCI
name "ich9-usb-uhci4", bus PCI
name "ich9-usb-uhci5", bus PCI
name "ich9-usb-uhci6", bus PCI
name "nec-usb-xhci", bus PCI
name "pci-ohci", bus PCI, desc "Apple USB Controller"
name "piix3-usb-uhci", bus PCI
name "piix4-usb-uhci", bus PCI
name "usb-ehci", bus PCI
name "vt82c686b-usb-uhci", bus PCI
```

实验四：查看qemu所能模拟的设备

```
Storage devices:
name "am53c974", bus PCI, desc "AMD Am53c974 PCscsi-PCI SCSI adapter"
name "dc390", bus PCI, desc "Tekram DC-390 SCSI adapter"
name "ich9-ahci", bus PCI, alias "ahci"
name "ide-cd", bus IDE, desc "virtual IDE CD-ROM"
name "ide-drive", bus IDE, desc "virtual IDE disk or CD-ROM (legacy)"
name "ide-hd", bus IDE, desc "virtual IDE disk"
name "isa-fdc", bus ISA
name "isa-ide", bus ISA
name "lsi53c810", bus PCI
name "lsi53c895a", bus PCI, alias "lsi"
name "megasas", bus PCI, desc "LSI MegaRAID SAS 1078"
name "nvme", bus PCI, desc "Non-Volatile Memory Express"
name "piix3-ide", bus PCI
name "piix3-ide-xen", bus PCI
name "piix4-ide", bus PCI
name "pvscsi", bus PCI
name "scsi-block", bus SCSI, desc "SCSI block device passthrough"
name "scsi-cd", bus SCSI, desc "virtual SCSI CD-ROM"
name "scsi-disk", bus SCSI, desc "virtual SCSI disk or CD-ROM (legacy)"
name "scsi-generic", bus SCSI, desc "pass through generic scsi device (/dev/sg*)"
name "scsi-hd", bus SCSI, desc "virtual SCSI disk"
name "usb-bot", bus usb-bus
name "usb-storage", bus usb-bus
name "usb-uas", bus usb-bus
name "vhost-scsi", bus virtio-bus
name "vhost-scsi-pci", bus PCI
name "virtio-9p-device", bus virtio-bus
name "virtio-9p-pci", bus PCI
name "virtio-blk-device", bus virtio-bus
name "virtio-blk-pci", bus PCI, alias "virtio-blk"
name "virtio-scsi-device", bus virtio-bus
name "virtio-scsi-pci", bus PCI
```

实验四：查看qemu所能模拟的设备

```
Network devices:
name "e1000", bus PCI, desc "Intel Gigabit Ethernet"
name "i82550", bus PCI, desc "Intel i82550 Ethernet"
name "i82551", bus PCI, desc "Intel i82551 Ethernet"
name "i82557a", bus PCI, desc "Intel i82557A Ethernet"
name "i82557b", bus PCI, desc "Intel i82557B Ethernet"
name "i82557c", bus PCI, desc "Intel i82557C Ethernet"
name "i82558a", bus PCI, desc "Intel i82558A Ethernet"
name "i82558b", bus PCI, desc "Intel i82558B Ethernet"
name "i82559a", bus PCI, desc "Intel i82559A Ethernet"
name "i82559b", bus PCI, desc "Intel i82559B Ethernet"
name "i82559c", bus PCI, desc "Intel i82559C Ethernet"
name "i82559er", bus PCI, desc "Intel i82559ER Ethernet"
name "i82562", bus PCI, desc "Intel i82562 Ethernet"
name "i82801", bus PCI, desc "Intel i82801 Ethernet"
name "ne2k_isa", bus ISA
name "ne2k_pci", bus PCI
name "pcnet", bus PCI
name "rtl8139", bus PCI
name "usb-bt-dongle", bus usb-bus
name "usb-net", bus usb-bus
name "virtio-net-device", bus virtio-bus
name "virtio-net-pci", bus PCI, alias "virtio-net"
name "vmxnet3", bus PCI, desc "VMWare Paravirtualized Ethernet v3"
```

实验四：查看qemu所能模拟的设备

```
Input devices:
name "ccid-card-passthru", bus ccid-bus, desc "passthrough smartcard"
name "ipoctal232", bus IndustryPack, desc "GE IP-Octal 232 8-channel RS-232 IndustryPack"
name "isa-parallel", bus ISA
name "isa-serial", bus ISA
name "pci-serial", bus PCI
name "pci-serial-2x", bus PCI
name "pci-serial-4x", bus PCI
name "tpci200", bus PCI, desc "TEWS TPCI200 IndustryPack carrier"
name "usb-braille", bus usb-bus
name "usb-ccid", bus usb-bus, desc "CCID Rev 1.1 smartcard reader"
name "usb-kbd", bus usb-bus
name "usb-mouse", bus usb-bus
name "usb-serial", bus usb-bus
name "usb-tablet", bus usb-bus
name "usb-wacom-tablet", bus usb-bus, desc "QEMU PenPartner Tablet"
name "virtconsole", bus virtio-serial-bus
name "virtio-serial-device", bus virtio-bus
name "virtio-serial-pci", bus PCI, alias "virtio-serial"
name "virtserialport", bus virtio-serial-bus

Display devices:
name "cirrus-vga", bus PCI, desc "Cirrus CLGD 54xx VGA"
name "isa-cirrus-vga", bus ISA
name "isa-vga", bus ISA
name "qxl", bus PCI, desc "Spice QXL GPU (secondary)"
name "qxl-vga", bus PCI, desc "Spice QXL GPU (primary, vga compatible)"
name "sga", bus ISA, desc "Serial Graphics Adapter"
name "VGA", bus PCI
name "vmware-svga", bus PCI
```

实验四：查看qemu所能模拟的设备

```
Sound devices:
name "AC97", bus PCI, desc "Intel 82801AA AC97 Audio"
name "adlib", bus ISA, desc "Yamaha YM3812 (OPL2)"
name "cs4231a", bus ISA, desc "Crystal Semiconductor CS4231A"
name "ES1370", bus PCI, desc "ENSONIQ AudioPCI ES1370"
name "gus", bus ISA, desc "Gravis Ultrasound GF1"
name "hda-duplex", bus HDA, desc "HDA Audio Codec, duplex (line-out, line-in)"
name "hda-micro", bus HDA, desc "HDA Audio Codec, duplex (speaker, microphone)"
name "hda-output", bus HDA, desc "HDA Audio Codec, output-only (line-out)"
name "ich9-intel-hda", bus PCI, desc "Intel HD Audio Controller (ich9)"
name "intel-hda", bus PCI, desc "Intel HD Audio Controller (ich6)"
name "sb16", bus ISA, desc "Creative Sound Blaster 16"
name "usb-audio", bus usb-bus

Misc devices:
name "i6300esb", bus PCI
name "ib700", bus ISA
name "isa-applesmc", bus ISA
name "isa-debug-exit", bus ISA
name "isa-debugcon", bus ISA
name "ivshmem", bus PCI
name "kvm-pci-assign", bus PCI, alias "pci-assign", desc "KVM-based PCI passthrough"
name "pc-testdev", bus ISA
name "pci-testdev", bus PCI, desc "PCI Test Device"
name "pvpanic", bus ISA
name "usb-redir", bus usb-bus
name "vfio-pci", bus PCI, desc "VFIO-based PCI device assignment"
name "virtio-balloon-device", bus virtio-bus
name "virtio-balloon-pci", bus PCI, alias "virtio-balloon"
name "virtio-rng-device", bus virtio-bus
name "virtio-rng-pci", bus PCI
name "xen-pci-passthrough", bus PCI, desc "Assign an host PCI device with Xen"
name "xen-platform", bus PCI, desc "XEN platform pci device"
```

2.4 QEMU-KVM: Images

- RAW Image
  - 格式简单，性能较好
  - 需要文件系统的支持才能支持sparse file，所以最好基于ext3
- 创建image

```
root@popsuper1982:/home/openstack/images# qemu-img create -f raw flat.img 10G
Formatting 'flat.img', fmt=raw size=10737418240
root@popsuper1982:/home/openstack/images# ls -lh flat.img
-rw-r--r-- 1 root root 10G Sep 23 09:46 flat.img
root@popsuper1982:/home/openstack/images# du -sh flat.img
0       flat.img
root@popsuper1982:/home/openstack/images# qemu-img info flat.img
image: flat.img
file format: raw
virtual size: 10G (10737418240 bytes)
disk size: 0
```

2.4 QEMU-KVM: Images

- 用dd产生image
  - 产生non sparse file
    - dd if=/dev/zero of=flat1.img bs=1024k count=1000
    - block size是1024k，共1000个block

```
root@popsuper1982:/home/openstack/images# dd if=/dev/zero of=flat1.img bs=1024k count=1000
1000+0 records in
1000+0 records out
1048576000 bytes (1.0 GB) copied, 4.59404 s, 228 MB/s
root@popsuper1982:/home/openstack/images# qemu-img info flat1.img
image: flat1.img
file format: raw
virtual size: 1.0G (1048576000 bytes)
disk size: 1.0G
```

  - 产生sparse file
    - dd if=/dev/zero of=flat2.img bs=1024k count=0 seek=2048
    - seek的意思是将文件的结尾设在那个地方

```
root@popsuper1982:/home/openstack/images# dd if=/dev/zero of=flat2.img bs=1024k count=0 seek=2048
0+0 records in
0+0 records out
0 bytes (0 B) copied, 0.00032575 s, 0.0 kB/s
root@popsuper1982:/home/openstack/images# qemu-img info flat2.img
image: flat2.img
file format: raw
virtual size: 2.0G (2147483648 bytes)
disk size: 0
```

2.4 QEMU-KVM: Images

- copy一个sparse文件
  - dd if=/dev/zero of=flat3.img bs=1024k count=1 seek=2048

```
root@popsuper1982:/home/openstack/images# dd if=/dev/zero of=flat3.img bs=1024k count=1 seek=2048
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00277215 s, 378 MB/s
root@popsuper1982:/home/openstack/images# qemu-img info flat3.img
image: flat3.img
file format: raw
virtual size: 2.0G (2148532224 bytes)
disk size: 1.0M
```

  - cp --sparse=never flat3.img flat3-never.img
    - 这样拷贝会占用整个2G空间

```
root@popsuper1982:/home/openstack/images# cp --sparse=never flat3.img flat3-never.img
root@popsuper1982:/home/openstack/images# qemu-img info flat3-never.img
image: flat3-never.img
file format: raw
virtual size: 2.0G (2148532224 bytes)
disk size: 2.0G
```

  - cp --sparse=always flat3.img flat3-always.img
    - 这样拷贝原来写入的1M的0也会被去掉

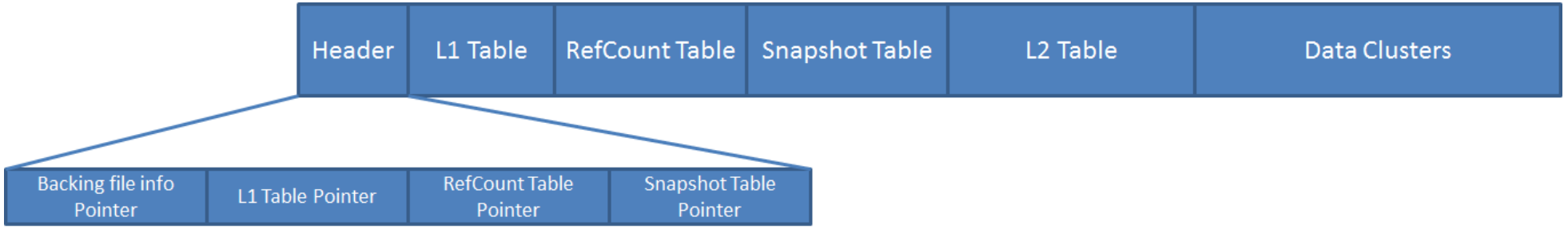
```
root@popsuper1982:/home/openstack/images# cp --sparse=always flat3.img flat3-always.img
root@popsuper1982:/home/openstack/images# qemu-img info flat3-always.img
image: flat3-always.img
file format: raw
virtual size: 2.0G (2148532224 bytes)
disk size: 0
```

  - cp --sparse=always flat3-never.img flat3-never-always.img
    - 这样拷贝原来的2G都会被清理掉

```
root@popsuper1982:/home/openstack/images# cp --sparse=always flat3-never.img flat3-never-always.img
root@popsuper1982:/home/openstack/images# qemu-img info flat3-never-always.img
image: flat3-never-always.img
file format: raw
virtual size: 2.0G (2148532224 bytes)
disk size: 0
```

2.4 QEMU-KVM: Images

- qcow2是动态的
  - 即便文件系统不支持sparse file，文件大小也很小
  - Copy on write
  - Snapshot
  - 压缩
  - 加密
- qcow2的格式



2.4 QEMU-KVM: Images

- 2-Level loopup
  - qcow2的数据是存储在data clusters里面的，每个cluster是512 byte sector
  - 为了能够管理这些cluster，qcow2保存了两层的Table，L1 table指向L2 Table，L2 Table管理data cluster.
  - 在image里面的offset会被解析成三部分，L1 Table Pointer先找L1，L1 Table Pointer+ offset[0]是L1中的一个entry，读出来便是L2 Table Pointer, L2 Table Pointer + offset[1]是L2中的一个entry，读出来便是data cluster pointer, data cluster pointer +offset[3]便是数据所在的位置。

2.4 QEMU-KVM: Images

- Copy-on-write
  - backing file就是基于这个原理的用处，一个qcow2的image可以保存另一个disk image的改变，而不影响另一个image
  - 创建backing file
    - qemu-img create -f qcow2 -o backing_file=./ubuntutest.qcow2 ubuntutest1.qcow2
    - 一开始新的image是空的，读取的内容都从老的image里面读取。
    - 当一个data cluster被写入，发生改变的时候，在新的image里面创建一个新的data cluster，这就是copy on write的意义。

```
root@popsuper1982:/home/openstack/images# qemu-img create -f qcow2 -o backing_file=./ubuntutest.qcow2 ubuntutest1.qcow2
Formatting 'ubuntutest1.qcow2', fmt=qcow2 size=5368709120 backing_file='./ubuntutest.qcow2' encryption=off cluster_size=65536 lazy_refcounts=off
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest1.qcow2
image: ubuntutest1.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 196K
cluster_size: 65536
backing file: ./ubuntutest.qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

2.4 QEMU-KVM: Images

- 基于ubuntutest1.img创建一个虚拟机
  - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest1.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
- 在虚拟机里面创建一个1G的non sparse file
  - dd if=/dev/zero of=flat1.img bs=1024k count=1000

```
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest1.qcow2
image: ubuntutest1.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 1.0G
cluster_size: 65536
backing file: ./ubuntutest.qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

2.4 QEMU-KVM: Images

- Convert: image格式之间可以转换
  - raw可以转换为qcow2
    - 创建一个raw image: dd if=/dev/zero of=flat.img bs=1024k count=1000
    - 进行转换： qemu-img convert -f raw -O qcow2 flat.img flat.qcow2
  - qcow2也可以转换为qcow2，转换的过程中，没用的data cluster就被去掉
    - qemu-img convert -f qcow2 -O qcow2 ubuntutest.img ubuntutest-convert.img

```
root@popsuper1982:/home/openstack/images# dd if=/dev/zero of=flat.img bs=1024k count=1000
1000+0 records in
1000+0 records out
1048576000 bytes (1.0 GB) copied, 4.91472 s, 213 MB/s
root@popsuper1982:/home/openstack/images# qemu-img info flat.img
image: flat.img
file format: raw
virtual size: 1.0G (1048576000 bytes)
disk size: 1.0G
root@popsuper1982:/home/openstack/images# qemu-img convert -f raw -O qcow2 flat.img flat.qcow2
root@popsuper1982:/home/openstack/images# qemu-img info flat.qcow2
image: flat.qcow2
file format: qcow2
virtual size: 1.0G (1048576000 bytes)
disk size: 196K
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

2.4 QEMU-KVM: Images

- 压缩compress
  - qemu-img convert -c -f qcow2 -O qcow2 ubuntutest.img ubuntutest-compress.qcow2

```
root@popsuper1982:/home/openstack/images# qemu-img convert -c -f qcow2 -O qcow2 ubuntutest.img ubuntutest-compress.qcow2
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest-compress.qcow2
image: ubuntutest-compress.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 452M
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

2.4 QEMU-KVM: Images

- 加密:
  - qemu-img convert -o encryption -f qcow2 -O qcow2 ubuntutest.qcow2 ubuntutest-encrypt.qcow2
  - 会要求设置密码
  - 从这个加密image启动一个虚拟机
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest-encrypt.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
    - 一开始虚拟机并不启动
    - 在monitor中输入cont
    - 需要输入密码后虚拟机才启动

```
root@popsuper1982:/home/openstack/images# qemu-img convert -o encryption -f qcow2 -O qcow2 ubuntutest.qcow2 ubuntutest-encrypt.qcow2
Disk image 'ubuntutest-encrypt.qcow2' is encrypted.
password:
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 -enable-kvm -name ubuntutest  -m 2048 -hda ubuntutest-encrypt.qcow2 -vnc :19 -net ni
c,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
QEMU 2.0.0 monitor - type 'help' for more information
(qemu) cont
ide0-hd0 (ubuntutest-encrypt.qcow2) is encrypted.
Password: ********
```

2.4 QEMU-KVM: Images

- 扩展
  - cp ubuntutest.qcow2 ubuntutest-enlarge.qcow2
  - qemu-img resize ubuntutest-enlarge.qcow2 +10G
  - 扩大的空间既不会被partition，也不会被format

```
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest-enlarge.qcow2
image: ubuntutest-enlarge.qcow2
file format: qcow2
virtual size: 15G (16106127360 bytes)
disk size: 512M
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

  - 启动虚拟机
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest-enlarge.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio

2.4 QEMU-KVM: Images

- Disk已经被扩大

```
root@ubuntutest:/home/openstack# fdisk -l

Disk /dev/sda: 16.1 GB, 16106127360 bytes
255 heads, 63 sectors/track, 1958 cylinders, total 31457280 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0001536d

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     6291455     3144704   83  Linux
/dev/sda2         6293502    10483711     2095105    5  Extended
/dev/sda5         6293504    10483711     2095104   82  Linux swap / Solaris
```

- 删除/dev/sda2和/dev/sda5
  - fdisk /dev/sda
  - 删除分区五：d 5
  - 删除分区二：d 2
  - 写入修改：w
  - reboot

2.4 QEMU-KVM: Images

- 扩展/dev/sda1分区到整块硬盘
  - fdisk /dev/sda
  - 删除根分区：d 1
  - 在原来根分区的位置创建新分区：n
  - 是一个主分区：p
  - 其他都选默认，最后写入修改：w
  - partprobe
- 扩展文件系统
  - resize2fs /dev/sda1

```
root@ubuntutest:/home/openstack# resize2fs /dev/sda1
resize2fs 1.42.9 (4-Feb-2014)
Filesystem at /dev/sda1 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/sda1 is now 3931904 blocks long.

root@ubuntutest:/home/openstack# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        15G  1.1G   13G   8% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            991M   12K  991M   1% /dev
tmpfs           201M  312K  200M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none           1002M     0 1002M   0% /run/shm
none            100M     0  100M   0% /run/user
```

2.5 QEMU-KVM: Snapshot

- snapshot也是copy on write的一种应用，和backing file有微妙的不同
- 有两种snapshot，一中时internal snapshot，一种是external snapshot
- internal snapshot是qcow2中的snapshot table所维护的snapshot，所有的snapshot都是在同一个文件中维护
- 创建一个虚拟机
  - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
- 在monitor中执行savevm后，ubuntutest.img会在image内部打一个snapshot

```
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest.qcow2
image: ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 752M
cluster_size: 65536
Snapshot list:
ID        TAG                 VM SIZE                DATE       VM CLOCK
1         vm-20140923183123      238M 2014-09-23 18:31:23   00:01:13.537
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

2.5 QEMU-KVM: Snapshot

- 原理是，当打一个snapshot后，会在snapshot table中建立一项，但是起初是空的，包含L1 table的一个复制，当L2 table或者data cluster改变的时候，则会将原来的数据复制一份，由snapshot的L1 table来维护，而原来的data cluster已经改变，在原地。
- 我们在/home/openstack下面创建一个文件夹
- loadvm vm-20140923183123
- 文件夹消失了
- 删除snapshot
  - delvm vm-20140923183123

2.5 QEMU-KVM: Snapshot

- external snapshot则往往采用上面的copy on write的方法
  - 当打snapshot的时候，将当前的image不再改变，创建一个新的image，以原来的image作为backing file，然后虚拟机使用新的image
  - 在monitor中，snapshot_blkdev ide0-hd0 ubuntutest-snapshot.img qcow2

```
(qemu) info block
ide0-hd0: ubuntutest.qcow2 (qcow2)

ide1-cd0: [not inserted]
    Removable device: not locked, tray closed

floppy0: [not inserted]
    Removable device: not locked, tray closed

sd0: [not inserted]
    Removable device: not locked, tray closed
(qemu) snapshot_blkdev ide0-hd0 ubuntutest-snapshot.img qcow2
Formatting 'ubuntutest-snapshot.img', fmt=qcow2 size=5368709120 backing_file='ubuntutest.qcow2' backing_fmt='qcow2' encryption=off cluster_size=6
5536 lazy_refcounts=off
(qemu) info block
ide0-hd0: ubuntutest-snapshot.img (qcow2)
    Backing file:     ubuntutest.qcow2 (chain depth: 1)

ide1-cd0: [not inserted]
    Removable device: not locked, tray closed

floppy0: [not inserted]
    Removable device: not locked, tray closed

sd0: [not inserted]
    Removable device: not locked, tray closed
```

  - 在HOST机器上多了一个文件

```
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest-snapshot.img
image: ubuntutest-snapshot.img
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 196K
cluster_size: 65536
backing file: ubuntutest.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

2.5 QEMU-KVM: Snapshot

- backing file可以是raw，也可以是qcow2，但是一旦打了snapshot，新的格式就是qcow2了。
- 两者很相似，稍微的不同是：
  - 对于internal snapshot, 刚打完snapshot的时候，原image集合是不变的，snapshot的集合是空的，接下来的操作，写入在原image，将不变的加入snapshot集合
  - 对于external snapshot，刚打完snapshot的时候，原image变成snapshot, snapshot集合是全集，新image是空的，接下来的操作，写入在新image，将改变的加入新image的集合。
- **qemu-img snapshot –c**
  - if the domain is offline and –disk-only was not specified
- **savevm**
  - if the domain is online and –disk-only was not specified
- **snapshot_blkdev**
  - if the domain is online and –disk-only is specified

2.6 QEMU-KVM: Network Block Device

- 网络块设备是通过NBD Server将虚拟块设备通过TCP/IP export出来，可以远程访问。
- NBD Server通常是qemu-nbd
- 方式一：可以提供unix socket
  - qemu-nbd -t -k /home/openstack/images/ubuntutest-nbd ubuntutest.img
  - 连接这个unix socket
  - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda nbd:unix:/home/openstack/images/ubuntutest-nbd -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=n -monitor stdio
- 方式二：普通的socket连接
  - qemu-nbd -t -p 1088 ubuntutest.qcow2
  - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda nbd:16.158.166.150:1088 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=n -monitor stdio

2.6 QEMU-KVM: Network Block Device

- 方式三：将image mount到一个network block device
  - 查看内核是否编译进去NBD

    #grep NBD /boot/config-3.13.0-24-generic
    CONFIG_BLK_DEV_NBD=m

  - 查看内核模块信息modinfo nbd

```
root@popsuper1982:/home/openstack/images# modinfo nbd
filename:       /lib/modules/3.13.0-24-generic/kernel/drivers/block/nbd.ko
license:        GPL
description:    Network Block Device
srcversion:     AF695984B8264FB5961650A
depends:
intree:         Y
vermagic:       3.13.0-24-generic SMP mod_unload modversions
signer:         Magrathea: Glacier signing key
sig_key:        00:A5:A6:57:59:DE:47:4B:C5:C4:31:20:88:0C:1B:94:A5:39:F4:31
sig_hashalgo:   sha512
parm:           nbds_max:number of network block devices to initialize (default: 16) (int)
parm:           max_part:number of partitions per device (default: 0) (int)
parm:           debugflags:flags for controlling debug output (int)
```

  - 查看内核模块是否加载lsmod | grep nbd
  - 如果没有加载modprobe nbd，也可以指定最多的partition: modprobe nbd max_part=16

2.6 QEMU-KVM: Network Block Device

- 加载后出现16个NBD

```
root@popsuper1982:/home/openstack/images# ls /dev/nbd*
/dev/nbd0  /dev/nbd10  /dev/nbd12  /dev/nbd14  /dev/nbd2  /dev/nbd4  /dev/nbd6  /dev/nbd8
/dev/nbd1  /dev/nbd11  /dev/nbd13  /dev/nbd15  /dev/nbd3  /dev/nbd5  /dev/nbd7  /dev/nbd9
```

- 查看哪个nbd device被使用：cat /proc/partitions

```
root@popsuper1982:/home/openstack/images# cat /proc/partitions
major minor  #blocks  name

   8        0  312571224 sda
   8        1  308588544 sda1
   8        2          1 sda2
   8        5    3980288 sda5
  11        0    1048575 sr0
```

2.6 QEMU-KVM: Network Block Device

- 将image付给一个network block device
  - qemu-nbd -c /dev/nbd0 ubuntutest.img
- 可以看到这个image里面有三个partition

```
root@popsuper1982:/home/openstack/images# ls /dev/nbd0*
/dev/nbd0  /dev/nbd0p1  /dev/nbd0p2  /dev/nbd0p5
root@popsuper1982:/home/openstack/images# fdisk -l /dev/nbd0

Disk /dev/nbd0: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders, total 10485760 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0001536d

     Device Boot      Start         End      Blocks   Id  System
/dev/nbd0p1   *        2048     6291455     3144704   83  Linux
/dev/nbd0p2         6293502    10483711     2095105    5  Extended
/dev/nbd0p5         6293504    10483711     2095104   82  Linux swap / Solaris
```

- Mount其中一个partition

```
root@popsuper1982:/home/openstack/images# mkdir ubuntutestnbd0p1
root@popsuper1982:/home/openstack/images# mount /dev/nbd0p1 ubuntutestnbd0p1/
root@popsuper1982:/home/openstack/images# ls ubuntutestnbd0p1/
bin  boot  dev  etc  home  initrd.img  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

2.6 QEMU-KVM: Network Block Device

- 修改结束后
  - umount ubuntutestnbd0p1
  - qemu-nbd -d /dev/nbd0

```
root@popsuper1982:/home/openstack/images# umount ubuntutestnbd0p1
root@popsuper1982:/home/openstack/images# qemu-nbd -d /dev/nbd0
/dev/nbd0 disconnected
```

2.6 QEMU-KVM: Network Block Device

- 有LVM的情况相对复杂
  - qemu-nbd -c /dev/nbd0 centos-5.8.new.qcow2

```
root@popsuper1982:/home/openstack/images# qemu-nbd -c /dev/nbd0 centos-5.8.new.qcow2
root@popsuper1982:/home/openstack/images# fdisk -l /dev/nbd0

Disk /dev/nbd0: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000f117e

     Device Boot      Start         End      Blocks   Id  System
/dev/nbd0p1   *          63      208844      104391   83  Linux
/dev/nbd0p2          208845    20964824    10377990   8e  Linux LVM
```

  - 发现里面有LVM，当然LVM不能作为整体访问，因为里面有Logic volume，都是单独成文件系统的
  - 查看LVM的信息

```
root@popsuper1982:/home/openstack/images# pvs
  PV         VG         Fmt  Attr PSize PFree
  /dev/nbd0p2 VolGroup00 lvm2 a-   9.88g    0
root@popsuper1982:/home/openstack/images# vgs
  VG         #PV #LV #SN Attr   VSize VFree
  VolGroup00   1   2   0 wzx-n- 9.88g    0
root@popsuper1982:/home/openstack/images# lvs
  Volume group VolGroup00 is exported
```

2.6 QEMU-KVM: Network Block Device

- Import这个volume group
  - vgimport VolGroup00
- 将这个volume group设为active
  - vgchange -ay VolGroup00

```
root@popsuper1982:/home/openstack/images# vgimport VolGroup00
  Volume group "VolGroup00" successfully imported
root@popsuper1982:/home/openstack/images# lvs
  LV       VG         Attr      LSize Pool Origin Data%  Move Log Copy%  Convert
  LogVol00 VolGroup00 -wi------ 5.97g
  LogVol01 VolGroup00 -wi------ 3.91g
root@popsuper1982:/home/openstack/images# vgchange -ay VolGroup00
  2 logical volume(s) in volume group "VolGroup00" now active
```

- Mount其中一个LV
  - mount /dev/VolGroup00/LogVol00 ubuntutestnbd0p1/

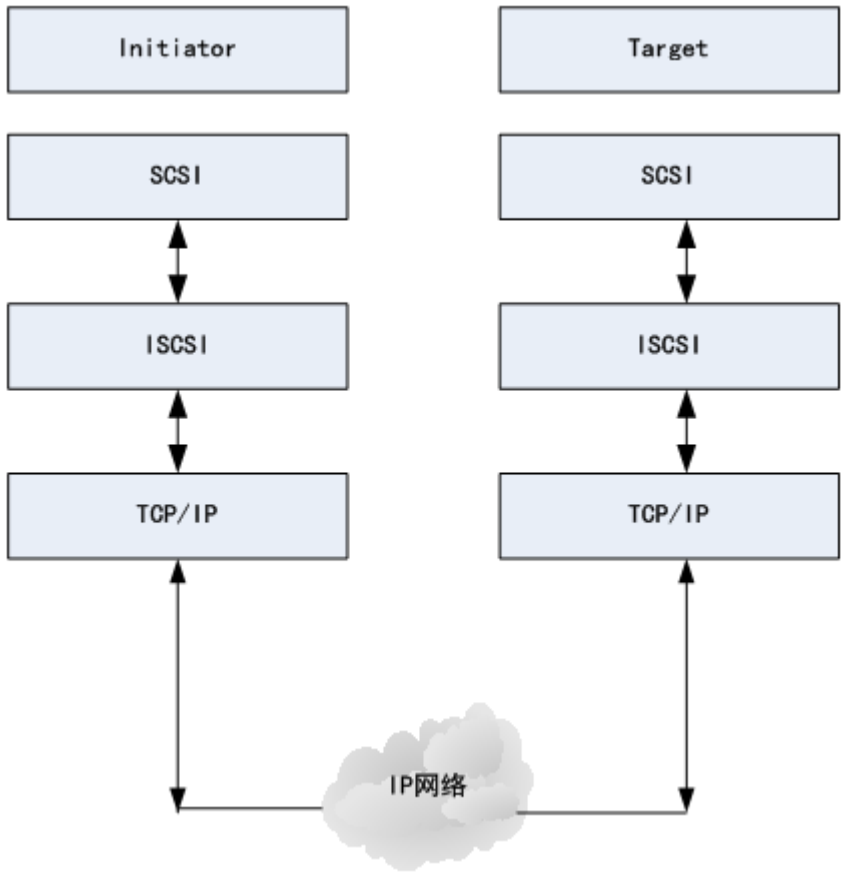
2.6 QEMU-KVM: Network Block Device

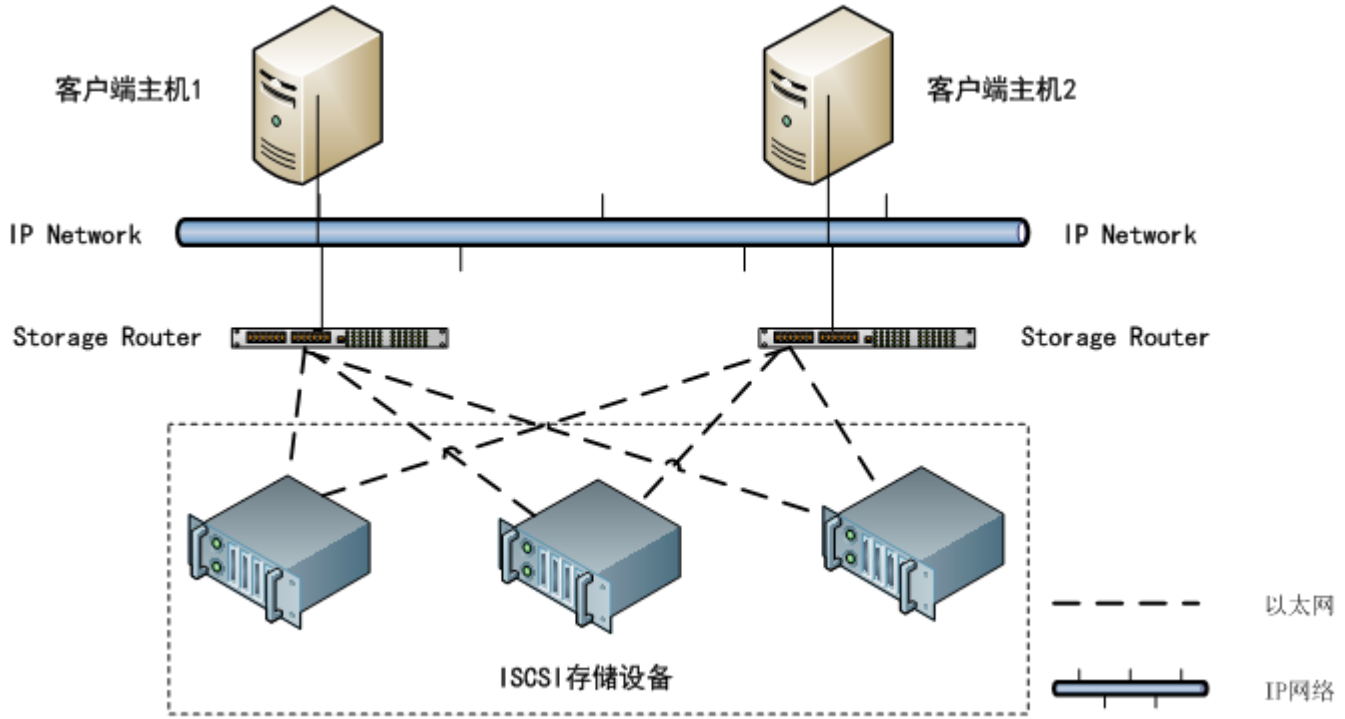
- 修改结束后
  - umount ubuntutestnbd0p1/
  - vgchange -an VolGroup00
  - vgexport VolGroup00
  - qemu-nbd -d /dev/nbd0

```
root@popsuper1982:/home/openstack/images# umount ubuntutestnbd0p1/
root@popsuper1982:/home/openstack/images# vgchange -an VolGroup00
  0 logical volume(s) in volume group "VolGroup00" now active
root@popsuper1982:/home/openstack/images# vgexport VolGroup00
  Volume group "VolGroup00" successfully exported
root@popsuper1982:/home/openstack/images# qemu-nbd -d /dev/nbd0
/dev/nbd0 disconnected
```

2.7 QEMU-KVM: 访问iSCSI

- iSCSI (Internet Small Computer Systems Interface)





2.7 QEMU-KVM: 访问iSCSI

- 在Server端：
  - 创建Physical Volume: pvcreate /dev/vdb
  - 创建Volume Group: vgcreate my-volume-group /dev/vdb
  - 创建Logical Volume: lvcreate -L 8G -n my_logical_volume my-volume-group

```
root@Nagios-Server:/home/openstack# fdisk -l

Disk /dev/vda: 21.5 GB, 21474836480 bytes
107 heads, 16 sectors/track, 24499 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00077a7b

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1            2048    41943039    20970496   83  Linux

Disk /dev/vdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/vdb doesn't contain a valid partition table
```

```
root@Nagios-Server:/home/openstack# lvdisplay
  --- Logical volume ---
  LV Path                /dev/my-volume-group/my_logical_volume
  LV Name                my_logical_volume
  VG Name                my-volume-group
  LV UUID                BiWNjf-Rciv-oxEl-cuUw-0kfp-LWcE-KdLm38
  LV Write Access        read/write
  LV Creation host, time Nagios-Server, 2014-09-24 09:38:32 -0400
  LV Status              available
  # open                 0
  LV Size                8.00 GiB
  Current LE             2048
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0
```

  - 将image写入LV:
    - qemu-img convert -O raw ubuntutest.qcow2 /dev/my-volume-group/my_logical_volume

2.7 QEMU-KVM: 访问iSCSI

- 在Server端
  - 用tgt查看iscsi target： tgtadm --mode target --op show
  - 创建一个iscsi target
    - tgtadm --lld iscsi --op new --mode target --tid 1 -T iqn.2014-09.org.openstack:my-iscsi-volume
  - 将Logic Volume加入刚才创建的iscsi target
    - tgtadm --lld iscsi --op new --mode logicalunit --tid 1 --lun 1 -b /dev/my-volume-group/my_logical_volume
  - 配置iscsi target监听链接
    - tgtadm --lld iscsi --op bind --mode target --tid 1 -I ALL

```
root@Nagios-Server:/home/openstack# netstat -tulpn | grep 3260
tcp        0      0 0.0.0.0:3260            0.0.0.0:*               LISTEN      10120/tgtd
tcp6       0      0 :::3260                 :::*                    LISTEN      10120/tgtd
```

```
root@Nagios-Server:/home/openstack# tgtadm --mode target --op show
Target 1: iqn.2014-09.org.openstack:my-iscsi-volume
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
    LUN information:
        LUN: 0
            Type: controller
            SCSI ID: IET     00010000
            SCSI SN: beaf10
            Size: 0 MB, Block size: 1
            Online: Yes
            Removable media: No
            Prevent removal: No
            Readonly: No
            SWP: No
            Thin-provisioning: No
            Backing store type: null
            Backing store path: None
            Backing store flags:
        LUN: 1
            Type: disk
            SCSI ID: IET     00010001
            SCSI SN: beaf11
            Size: 8590 MB, Block size: 512
            Online: Yes
            Removable media: No
            Prevent removal: No
            Readonly: No
            SWP: No
            Thin-provisioning: No
            Backing store type: rdwr
            Backing store path: /dev/my-volume-group/my_logical_volume
            Backing store flags:
    Account information:
    ACL information:
```

2.7 QEMU-KVM: 访问iSCSI

- 在客户端
  - discover这个iscsi target
    - iscsiadm --mode discovery --type sendtargets --portal 192.168.100.200

```
root@popsuper1982:/home/openstack/images# iscsiadm --mode discovery --type sendtargets --portal 192.168.100.200
192.168.100.200:3260,1 iqn.2014-09.org.openstack:my-iscsi-volume
```

  - 连接那个iscsi target
    - iscsiadm --mode node --targetname iqn.2014-09.org.openstack:my-iscsi-volume --portal 16.158.166.197:3260192.168.100.200:3260 –login

```
root@popsuper1982:/home/openstack/images# iscsiadm --mode node --targetname iqn.2014-09.org.openstack:my-iscsi-volume --portal 1
192.168.100.200:3260 --login
Logging in to [iface: default, target: iqn.2014-09.org.openstack:my-iscsi-volume, portal: 192.168.100.200,3260] (multiple)
Login to [iface: default, target: iqn.2014-09.org.openstack:my-iscsi-volume, portal: 192.168.100.200,3260] successful.
```

- Qemu可以直接连接/dev/sdb

```
root@popsuper1982:/home/openstack/images# fdisk -l

Disk /dev/sda: 320.1 GB, 320072933376 bytes
255 heads, 63 sectors/track, 38913 cylinders, total 625142448 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000162d4

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048   617179135   308588544   83  Linux
/dev/sda2       617181182   625141759     3980289    5  Extended
/dev/sda5       617181184   625141759     3980288   82  Linux swap / Solaris

Disk /dev/sdb: 8589 MB, 8589934592 bytes
64 heads, 32 sectors/track, 8192 cylinders, total 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0001536d

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1   *        2048     6291455     3144704   83  Linux
/dev/sdb2         6293502    10483711     2095105    5  Extended
/dev/sdb5         6293504    10483711     2095104   82  Linux swap / Solaris
```

2.7 QEMU-KVM: 访问iSCSI

- 在客户端
  - Qemu也可以直接连接iSCSI
  - 但是需要编译的时候，启动--enable-libiscsi，将libiscsi编译进去
  - 然而默认的ubuntu下面的没有编译进去
  - ldd /usr/bin/qemu-system-x86_64

```
root@popsuper1982:/# ldd /usr/bin/qemu-system-x86_64 | grep iscsi
root@popsuper1982:/#
```

```
root@popsuper1982:/# ldd /usr/bin/qemu-system-x86_64
        linux-vdso.so.1 =>  (0x00007fffac951000)
        libaio.so.1 => /lib/x86_64-linux-gnu/libaio.so.1 (0x00007f1b3db8e000)
        libcurl-gnutls.so.4 => /usr/lib/x86_64-linux-gnu/libcurl-gnutls.so.4 (0x00007f1b3d92d000)
        librados.so.2 => /usr/lib/x86_64-linux-gnu/librados.so.2 (0x00007f1b3ca16000)
        librbd.so.1 => /usr/lib/x86_64-linux-gnu/librbd.so.1 (0x00007f1b3c72a000)
        libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f1b3c511000)
        librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f1b3c308000)
        libglib-2.0.so.0 => /lib/x86_64-linux-gnu/libglib-2.0.so.0 (0x00007f1b3c000000)
        libutil.so.1 => /lib/x86_64-linux-gnu/libutil.so.1 (0x00007f1b3bdfd000)
        libbluetooth.so.3 => /usr/lib/x86_64-linux-gnu/libbluetooth.so.3 (0x00007f1b3bbdf000)
        libncurses.so.5 => /lib/x86_64-linux-gnu/libncurses.so.5 (0x00007f1b3b9bc000)
        libtinfo.so.5 => /lib/x86_64-linux-gnu/libtinfo.so.5 (0x00007f1b3b793000)
        libbrlapi.so.0.6 => /lib/x86_64-linux-gnu/libbrlapi.so.0.6 (0x00007f1b3b586000)
```

- ISCSI Logout
  - iscsiadm --mode node --targetname iqn.2014-09.org.openstack:my-iscsi-volume --portal 192.168.100.200:3260 --logout

```
root@popsuper1982:/home/openstack/images# iscsiadm --mode node --targetname iqn.2014-09.org.openstack:my-iscsi-volume --portal
192.168.100.200:3260 --logout
Logging out of session [sid: 1, target: iqn.2014-09.org.openstack:my-iscsi-volume, portal: 192.168.100.200,3260]
Logout of [sid: 1, target: iqn.2014-09.org.openstack:my-iscsi-volume, portal: 192.168.100.200,3260] successful.
```

2.8 QEMU-KVM:网络虚拟化

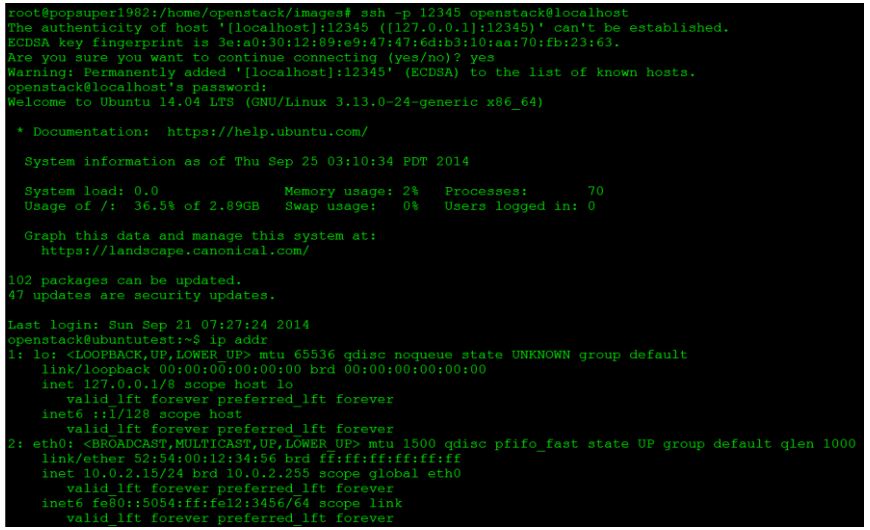
- 网卡配置(Guest角度)：
  - -net nic[,vlan=n][,macaddr=mac][,model=type][,Name=str][,addr=str][,vectors=v]
  - VLAN和802.1.q一点关系都没有，就是virtual hub的概念，vlan参数表示连接在哪个virtual hub上
  - Virtual hub在同一个qemu进程中有效，属于同一个virtual hub的网卡可以彼此收到包
  - model有多种：
    - qemu-system-x86_64 -net nic,model=?

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 -net nic,model=?
qemu: Supported NIC models: ne2k_pci,i82551,i82557b,i82559er,rtl8139,e1000,pcnet,virtio
```

- 网络模式(Host角度)
  - -net user/socket/tap

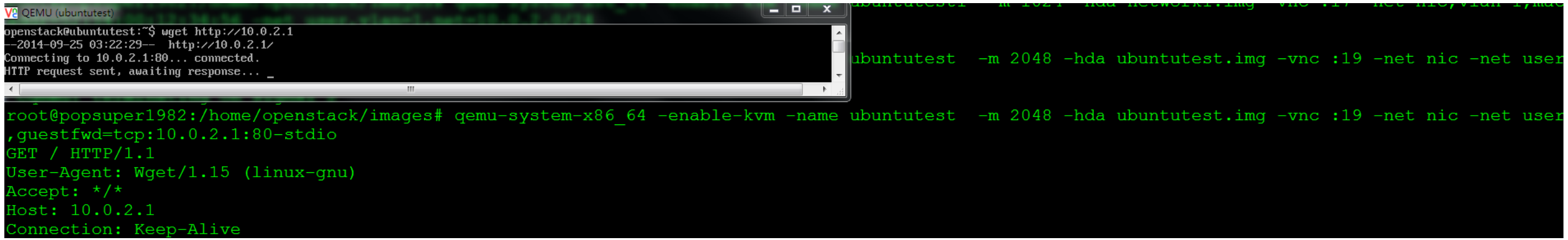
2.8 QEMU-KVM:网络虚拟化

- 网络模式User Network： 如何对外通信？
  - hostfwd=[tcp|udp]:[hostaddr]:hostport-[guestaddr]:guestport
    - 所有去host上的hostport的包都会被转发给guest的guestport
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.img -vnc :19 -net nic -net user,hostfwd=tcp::12345-:22
    - 将在host机器上到端口12345的包转发给guest机器的22端口
    - 从host上ssh，可以进入虚拟机
  - 在monitor中也可以用下面的命令添加和删除hostfwd
    - hostfwd_add [VLAN_ID name] [tcp | udp]: [hostaddr]: host-port [guestaddr]: guest port
    - hostfwd_remove [VLAN_ID name] [tcp | udp]: [hostaddr]: host port



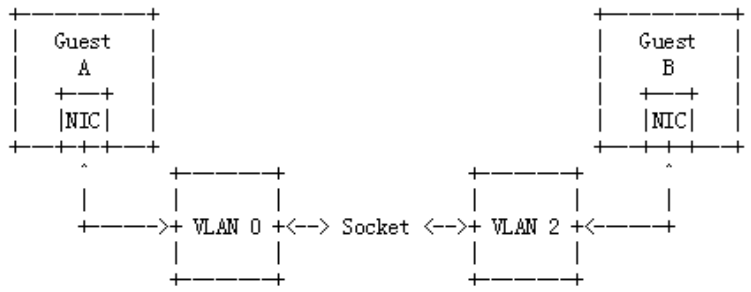
2.8 QEMU-KVM:网络虚拟化

- 网络模式User Network
  - guestfwd=[tcp]:server:port-dev
  - guestfwd=[tcp]:server:port-cmd:command
    - 在guest里面将连接server:port的包转发给一个设备，或者执行一个命令
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.img -vnc :19 -net nic -net user,guestfwd=tcp:10.0.2.1:80-stdio
    - 在guest里面运行wget http://10.0.2.1/index.html的时候，出现下面的输出



2.8 QEMU-KVM:网络虚拟化

- Socket：虚拟机之间的交互
  - 可以使用TCP，一对一交互
    - -net socket[,vlan=n][,name=name][,fd=h] [,listen=[host]:port][,connect=host:port]
    - qemu-system-x86_64 -enable-kvm -name ubuntutest1 -m 1024 -hda network1.img -vnc :17 -net nic,macaddr=52:54:00:12:34:56 -net socket,listen=:1234
    - qemu-system-x86_64 -enable-kvm -name ubuntutest2 -m 1024 -hda network2.img -vnc :18 -net nic,macaddr=52:54:00:12:34:57 -net socket,connect=127.0.0.1:1234



QEMU (ubuntutest1)

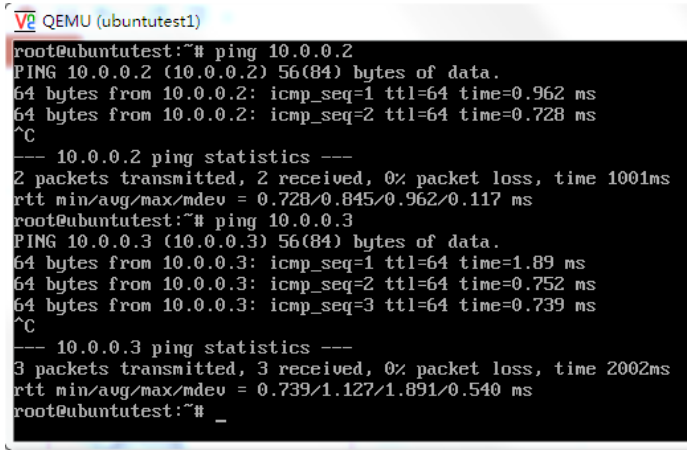
```
root@ubuntutest:/home/openstack# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe12:3456/64 scope link
       valid_lft forever preferred_lft forever
root@ubuntutest:/home/openstack# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.892 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.707 ms
^C
```

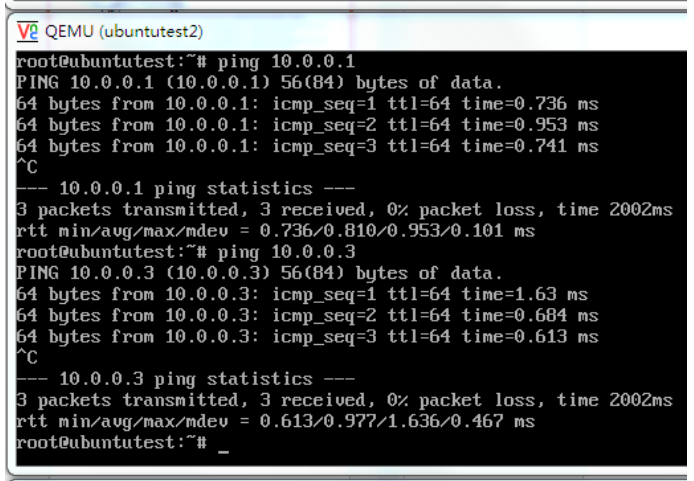
QEMU (ubuntutest2)

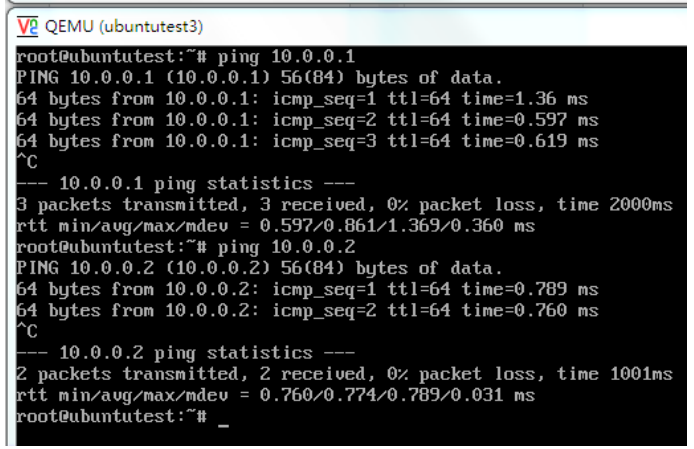
```
root@ubuntutest:/home/openstack# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:12:34:57 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/24 brd 10.0.0.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe12:3457/64 scope link
       valid_lft forever preferred_lft forever
root@ubuntutest:/home/openstack# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.774 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.730 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.648 ms
```

2.8 QEMU-KVM:网络虚拟化

- Socket：虚拟机之间的交互
  - 可以使用UDP，Multicast交互
    - -net socket[,vlan=n][,name=name][,fd=h][,mcast=maddr:port[,localaddr=addr]]
    - qemu-system-x86_64 -enable-kvm -name ubuntutest1 -m 1024 -hda network1.img -vnc :17 -net nic,macaddr=52:54:00:12:34:56 -net socket,mcast=230.0.0.1:1234
    - qemu-system-x86_64 -enable-kvm -name ubuntutest2 -m 1024 -hda network2.img -vnc :18 -net nic,macaddr=52:54:00:12:34:57 -net socket,mcast=230.0.0.1:1234
    - qemu-system-x86_64 -enable-kvm -name ubuntutest3 -m 1024 -hda network3.img -vnc :19 -net nic,macaddr=52:54:00:12:34:58 -net socket,mcast=230.0.0.1:1234

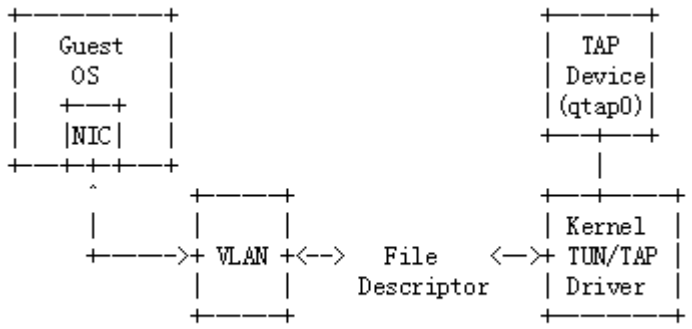






2.8 QEMU-KVM:网络虚拟化

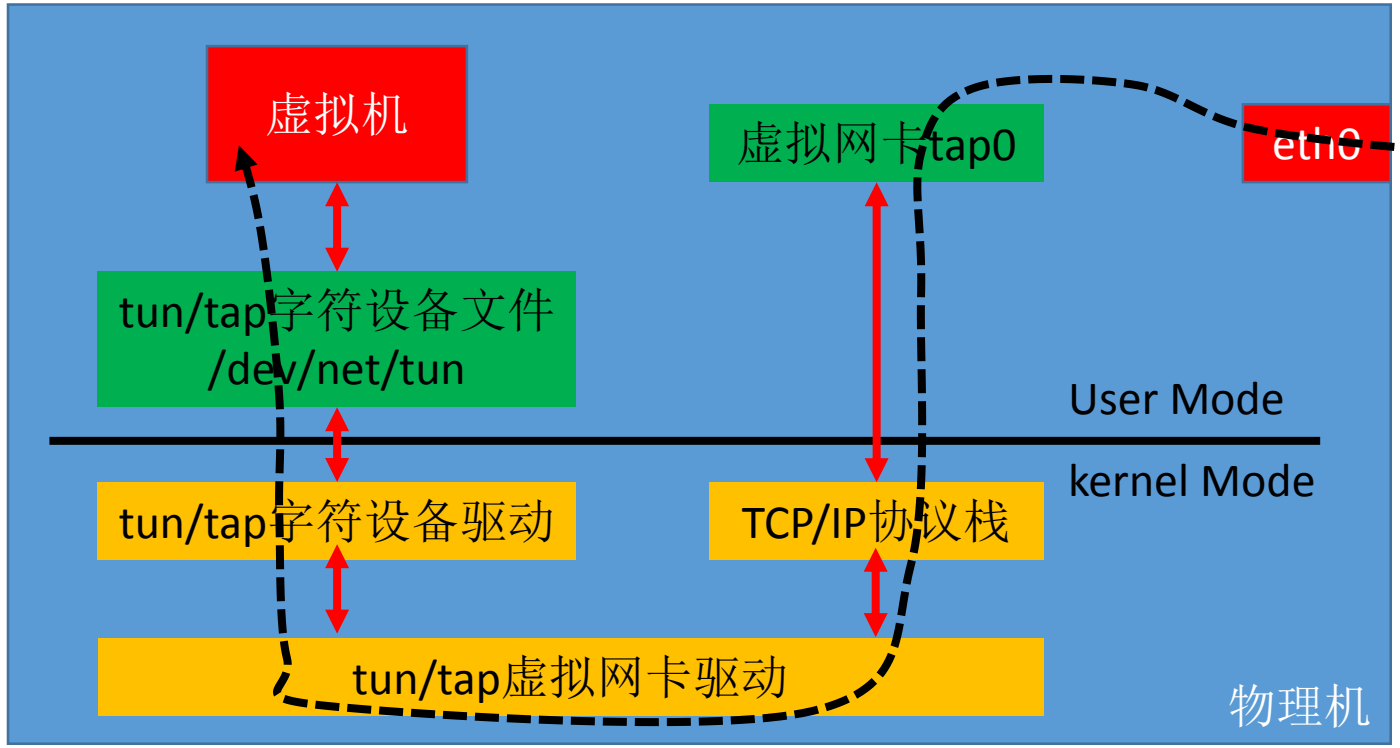
- Tap/Tun Device
  - 将guest system的网络和host system的网络连在一起。
  - 通过TUN/TAP adapter，会生成一个在host system上的虚拟网卡tap
  - tun建立了point to point的网络设备，使得guest system的网卡和tap虚拟网卡成为一对
  - 从而guest system的所有网络包，host system都能收到。




```
$> qemu -net nic -net tap,ifname=qtap0 ...
```

2.8 QEMU-KVM:网络虚拟化

- Tap/Tun Device
  - Tun/tap驱动程序中包含两个部分，一部分是字符设备驱动，还有一部分是网卡驱动部分

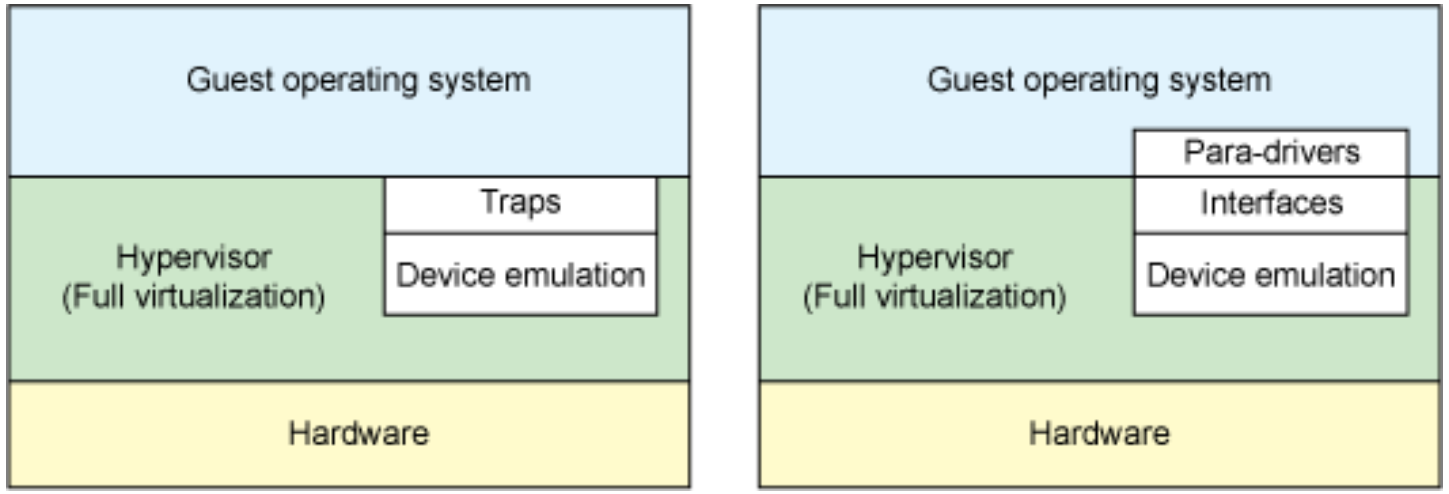


- 虚拟机将网络包通过字符设备写入/dev/net/tun
- 字符设备驱动将数据包写入虚拟网卡驱动
- 虚拟网卡驱动将包通过TCP/IP协议栈写给Host上的虚拟网卡tap0
- 在HOST上通过路由规则，包从eth0出去

```
root@popsuper1982:/home/openstack# ps aux | grep qemu | grep tap0
root     11688  1.7  8.2 2553192 316448 pts/8  Sl+  05:12   0:18 qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda u
c :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=n -monitor stdio
root@popsuper1982:/home/openstack# lsof -p 11688 | grep tun
qemu-syst 11688 root    8u   CHR   10,200      0t145      9273 /dev/net/tun
```

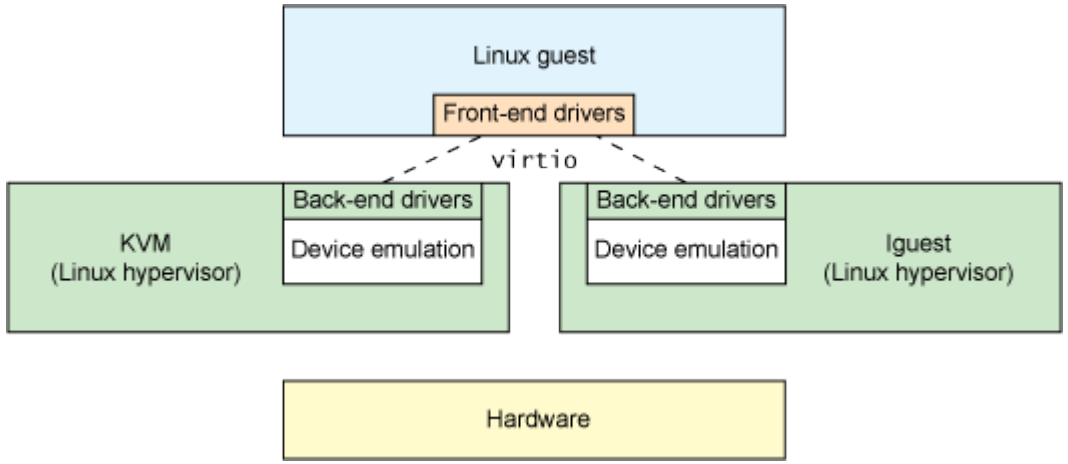
2.9 QEMU-KVM:半虚拟化设备virtio

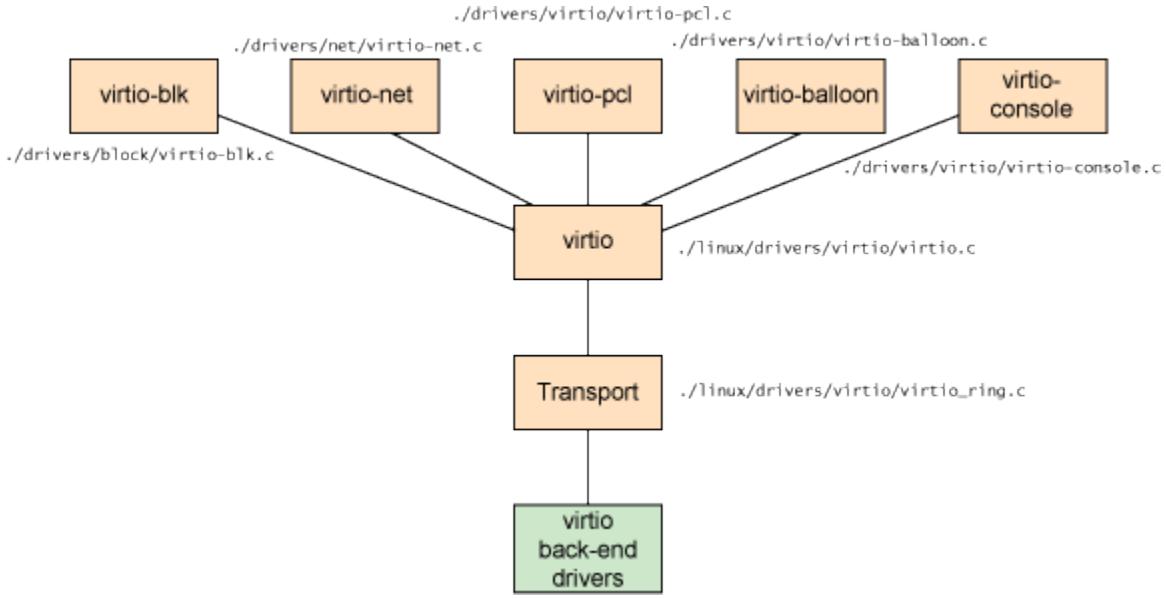
- 为了提高内存，硬盘，网络的性能，需要支持半虚拟化
- 在全虚拟化状态下，Guest OS不知道自己是虚拟机，于是像发送普通的IO一样发送数据，被Hypervisor拦截，转发给真正的硬件
- 在半虚拟化状态下，Guest需要安装半虚拟化驱动，Guest OS知道自己是虚拟机，所以数据直接发送给半虚拟化设备，经过特殊处理，发送给真正的硬件
- 半虚拟化驱动：virtio, Vmware Tools



2.9 QEMU-KVM:半虚拟化设备virtio

- 然而要虚拟的设备多种多样，要支持的硬件多种多样，需要统一的接口，virtio
  - 不同的虚拟设备和不同的虚拟机可以有不同的前端驱动
  - 不同的硬件设备可以有不同的后端驱动
  - 两者之间的交互遵循virtio的标准
    - virtio层是虚拟队列接口，virtio-net网络驱动程序使用两个虚拟队列（一个用于接收，另一个用于发送），而virtio-blk块驱动程序仅使用一个虚拟队列。
    - Transport(virtio-ring)实现了环形缓冲区(ring buffer),用于保存前端驱动和后端处理程序执行的信息，并且它可以一次性保存前端驱动的多次I/O请求，并且交由后端去动去批量处理





2.9 QEMU-KVM:半虚拟化设备virtio

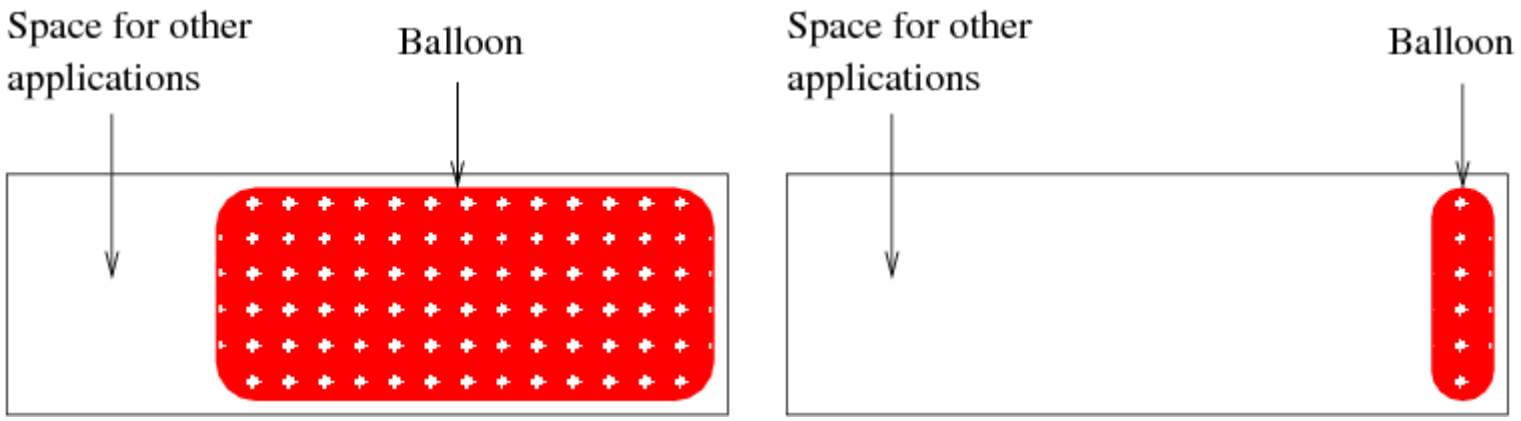
- Memory Ballooning (virtio_balloon)
  - memory ballooning可以动态调整guest的内存的大小
  - 如果有-m参数，则向更大的内存调整时无效的，但是可以往小的里面调整
  - 启动虚拟机：
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -balloon virtio -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=n -monitor stdio

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -balloon virtio -hda ubuntutest.qcow2 -vnc :19 -
net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=n -monitor stdio
QEMU 2.0.0 monitor - type 'help' for more information
(qemu) info balloon
balloon: actual=2048
(qemu) balloon 4096
(qemu) info balloon
balloon: actual=2048
(qemu) balloon 1024
(qemu) info balloon
balloon: actual=1024
```

```
root@ubuntutest:/home/openstack# free -h
             total       used       free     shared    buffers     cached
Mem:          2.0G       160M       1.8G       332K        16M        91M
-/+ buffers/cache:        52M       1.9G
Swap:         2.0G         0B       2.0G
root@ubuntutest:/home/openstack# free -h
             total       used       free     shared    buffers     cached
Mem:          978M       160M       817M       332K        16M        91M
-/+ buffers/cache:        51M       926M
Swap:         2.0G         0B       2.0G
```

2.9 QEMU-KVM:半虚拟化设备virtio

- 有了balloon，如果guest需要更多的RAM，则可以给它增加一些内存。
- 如果guest不需要那么多内存，可以通过balloon从中拿出一部分内存。这个过程，不需要启停虚拟机。
- balloon driver像是一个特殊的process，运行在guest机器上，它可以扩大自己的内存，使得其他的应用程序的内存减少，也可以缩小内存，使得其他的应用程序内存增加。



2.9 QEMU-KVM:半虚拟化设备virtio

- guest中的balloon driver通过virtio channel和host进行交互，接收host发来的命令，比如发来的命令式减少内存，则balloon driver就扩大它的内存占有量。
- 然后balloon driver将自己占有的内存交回给host，使得host有了更多的内存。
- 对于libvirt而言，有currentMemory和maxMemory两种概念，maxMemory就是-m参数设定的，currentMemory就是balloon设定的。

2.9 QEMU-KVM:半虚拟化设备virtio

- 硬盘虚拟化virtio_blk
  - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -balloon virtio -drive file=ubuntutest.qcow2,if=virtio -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=n -monitor stdio
  - 使用virtio_blk驱动的硬盘显示为/dev/vda，使用IDE硬盘显示为/dev/had，使用SATA硬盘显示/dev/sda
  - virtio-blk-data-plane：进一步提高性能，每个device单独的线程
    - 仅支持raw disk
    - 不支持storage migration

```
root@ubuntutest:/home/openstack# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Cirrus Logic GD 5446
00:03.0 Ethernet controller: Red Hat, Inc Virtio network device
00:04.0 Unclassified device [00ff]: Red Hat, Inc Virtio memory balloon
00:05.0 SCSI storage controller: Red Hat, Inc Virtio block device
```

2.9 QEMU-KVM:半虚拟化设备virtio

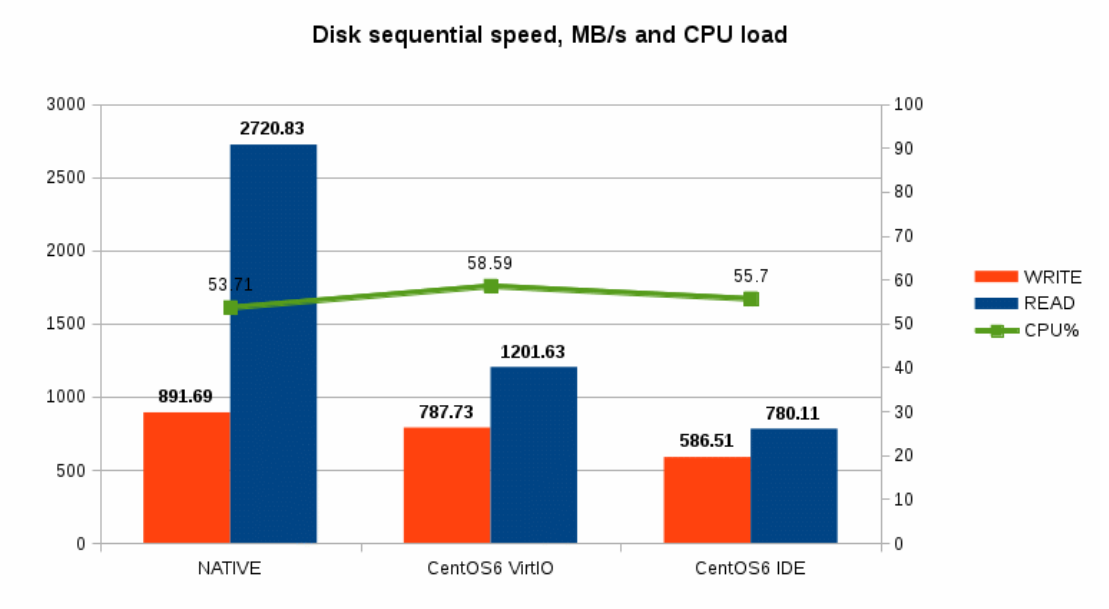
- 网络设备虚拟化virtio_net
  - ethtool -i eth0
  - TSO是通过网络设备进行TCP段的分割，从而来提高网络性能的一种技术
  - GSO(Generic Segmentation Offload) 应用于其他的传输层协议，如TCPv6，UDP
  - 应用层可以使用ethtool -K eth0 tso off|on命令对支持TSO特性的网络设备进行TSO功能的关闭和启用
  - 使用virtio_net性能低，可尝试关闭这两个选项

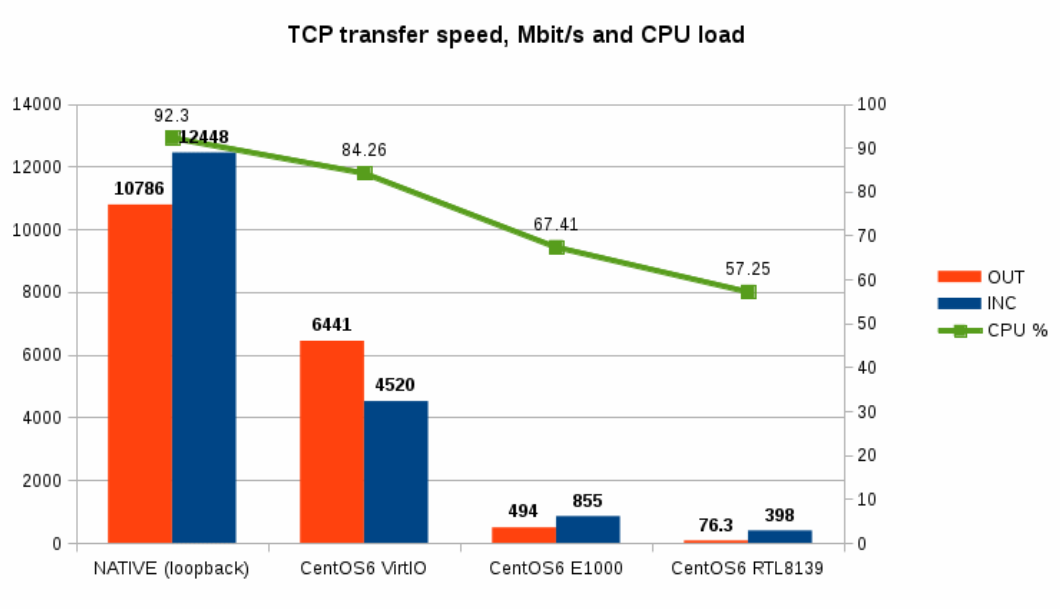
```
root@ubuntutest:/home/openstack# ethtool -i eth0
driver: virtio_net
version: 1.0.0
firmware-version:
bus-info: 0000:00:03.0
supports-statistics: no
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

```
root@ubuntutest:/home/openstack# ethtool -k eth0
Features for eth0:
rx-checksumming: off [fixed]
tx-checksumming: off
        tx-checksum-ipv4: off [fixed]
        tx-checksum-ip-generic: off [fixed]
        tx-checksum-ipv6: off [fixed]
        tx-checksum-fcoe-crc: off [fixed]
        tx-checksum-sctp: off [fixed]
scatter-gather: off
        tx-scatter-gather: off [fixed]
        tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: off
        tx-tcp-segmentation: off [fixed]
        tx-tcp-ecn-segmentation: off [fixed]
        tx-tcp6-segmentation: off [fixed]
udp-fragmentation-offload: off [fixed]
generic-segmentation-offload: off [requested on]
generic-receive-offload: on
large-receive-offload: off [fixed]
rx-vlan-offload: off [fixed]
tx-vlan-offload: off [fixed]
ntuple-filters: off [fixed]
receive-hashing: off [fixed]
highdma: on [fixed]
rx-vlan-filter: on [fixed]
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: off [fixed]
tx-ipip-segmentation: off [fixed]
tx-sit-segmentation: off [fixed]
tx-udp_tnl-segmentation: off [fixed]
tx-mpls-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off
loopback: off [fixed]
rx-fcs: off [fixed]
rx-all: off [fixed]
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
l2-fwd-offload: off [fixed]
```

2.9 QEMU-KVM:半虚拟化设备virtio

- 性能比较





2.10 QEMU-KVM: Migration

- 在qemu里面Live Migration是通过monitor进行的
- 方法一：使用共享存储，NFS, NBD, SAN
  - 将一台机器作为共享存储NBD
    - 16.158.166.150
    - qemu-nbd -t -p 1234 --share=2 ubuntutest.qcow2
  - 源机器和目标机器创建相同的网络环境，参考2.1节
  - 在源机器上启动虚拟机
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda nbd:16.158.166.150:1234 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
  - 在目标机器上启动虚拟机，监听migration端口
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda nbd:16.158.166.150:1234 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio -incoming tcp:0:1235

2.10 QEMU-KVM: Migration

- 方法一：使用共享存储， NFS, NBD, SAN
  - 用VNC连接到源机器的VM中，ping网关192.168.57.1
  - 在源机器的monitor中启动migration
    - migrate -d tcp:16.158.166.197:1235
  - 查看迁移情况
    - info migrate
    - 应该是active的状态
  - 用VNC连接到目标机器的VM中，发现仍在ping网关
  - 查看迁移情况
    - info migrate
    - 应该是completed的状态
  - 迁移成功

```
(qemu) migrate -d tcp:16.158.166.197:1235
(qemu) info migrate
capabilities: xbzrle: off rdma-pin-all: off auto-converge: off zero-blocks: off
Migration status: active
total time: 4235 milliseconds
expected downtime: 30 milliseconds
setup: 11 milliseconds
transferred ram: 48738 kbytes
throughput: 94.01 mbps
remaining ram: 1207244 kbytes
total ram: 2106060 kbytes
duplicate: 212813 pages
skipped: 0 pages
normal: 11890 pages
normal bytes: 47560 kbytes
(qemu) info migrate
capabilities: xbzrle: off rdma-pin-all: off auto-converge: off zero-blocks: off
Migration status: completed
total time: 22969 milliseconds
downtime: 2 milliseconds
setup: 11 milliseconds
transferred ram: 264046 kbytes
throughput: 94.75 mbps
remaining ram: 0 kbytes
total ram: 2106060 kbytes
duplicate: 464794 pages
skipped: 0 pages
normal: 64863 pages
normal bytes: 259452 kbytes
```

2.10 QEMU-KVM: Migration

- 方法二：连带migrate硬盘的迁移
  - 查看源机器上的image的情况
    - qemu-img info ubuntutest.qcow2

```
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest.qcow2
image: ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 754M
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

  - 在目标机器上，创建一个大小相同的空image
    - qemu-img create -f qcow2 ubuntutest.qcow2 5G

```
root@OCTO-BJ-Openstack:/home/cliu8/images# qemu-img create -f qcow2 ubuntutest.qcow2 5G
Formatting 'ubuntutest.qcow2', fmt=qcow2 size=5368709120 encryption=off cluster_size=65536 lazy_refcounts=off
root@OCTO-BJ-Openstack:/home/cliu8/images# qemu-img info ubuntutest.qcow2
image: ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 196K
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

2.10 QEMU-KVM: Migration

- 方法二：连带migrate硬盘的迁移
  - 在源机器上启动虚拟机
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
  - 在目标机器上启动虚拟机migrate监听
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio -incoming tcp:0:1235
  - 开始迁移
    - migrate -b tcp:16.158.166.197:1235

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=vi
rtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
QEMU 2.0.0 monitor - type 'help' for more information
(qemu) migrate -b tcp:16.158.166.197:1235
Completed 22 %
```

```
root@OCTO-BJ-Openstack:/home/cliu8/images# qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=v
irtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio -incoming tcp:0:1235
QEMU 2.0.0 monitor - type 'help' for more information
(qemu) Receiving block device images
Completed 100 %
```

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=vi
rtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
QEMU 2.0.0 monitor - type 'help' for more information
(qemu) migrate -b tcp:16.158.166.197:1235
Completed 99 %
(qemu) info migrate
capabilities: xbzrle: off rdma-pin-all: off auto-converge: off zero-blocks: off
Migration status: completed
total time: 508967 milliseconds
downtime: 2 milliseconds
setup: 11 milliseconds
transferred ram: 526212 kbytes
throughput: 99.74 mbps
remaining ram: 0 kbytes
total ram: 2106060 kbytes
duplicate: 467978 pages
skipped: 0 pages
normal: 130269 pages
normal bytes: 521076 kbytes
```

```
root@OCTO-BJ-Openstack:/home/cliu8/images# qemu-img info ubuntutest.qcow2
image: ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 5.0G
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

2.11 QEMU-KVM: Monitoring and Debugging

- 对Monitor的访问
  - [Ctrl] + [Alt] + [2]
  - -monitor stdio
  - 可以通过TCP连接：
    - -monitor tcp::4444,server,nowait
    - 通过telnet可以访问： telnet localhost 4444

```
root@popsuper1982:/home/openstack/images# telnet localhost 4444
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
QEMU 2.0.0 monitor - type 'help' for more information
(qemu) info block
info block
ide0-hd0: ubuntutest.qcow2 (qcow2)
```

  - 可以通过character device： -chardev stdio,id=meinmonitor -mon chardev=meinmonitor

2.11 QEMU-KVM: Monitoring and Debugging

- 访问The QEMU Monitor Protocol (QMP)：
  - 通过stdio
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -qmp stdio
    - 首先运行{"execute": "qmp_capabilities"}
    - 然后运行{“execute”: “query-commands”}查看所有的命令

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=vi
rtio -net tap,ifname=tap0,script=no,downscript=no -qmp stdio
{"QMP": {"version": {"qemu": {"micro": 0, "minor": 0, "major": 2}, "package": " (Debian 2.0.0+dfsg-2ubuntu1.2)"}, "capabilities": []}}
{"execute": "qmp_capabilities" }
{"return": {}}
{"execute": "query-commands"}
{"return": [{"name": "query-named-block-nodes"}, {"name": "blockdev-add"}, {"name": "query-rx-filter"}, {"name": "chardev-remove"}, {"name": "char
dev-add"}, {"name": "query-tpm-types"}, {"name": "query-tpm-models"}, {"name": "query-tpm"}, {"name": "query-target"}, {"name": "query-cpu-definit
ions"}, {"name": "query-machines"}, {"name": "device-list-properties"}, {"name": "qom-list-types"}, {"name": "change-vnc-password"}, {"name": "nbd
-server-stop"}, {"name": "nbd-server-add"}, {"name": "nbd-server-start"}, {"name": "qom-get"}, {"name": "qom-set"}, {"name": "qom-list"}, {"name":
 "query-block-jobs"}, {"name": "query-balloon"}, {"name": "query-migrate-capabilities"}, {"name": "migrate-set-capabilities"}, {"name": "query-mig
rate"}, {"name": "query-command-line-options"}, {"name": "query-uuid"}, {"name": "query-name"}, {"name": "query-spice"}, {"name": "query-vnc"}, {"
name": "query-mice"}, {"name": "query-status"}, {"name": "query-kvm"}, {"name": "query-pci"}, {"name": "query-iothreads"}, {"name": "query-cpus"},
 {"name": "query-blockstats"}, {"name": "query-block"}, {"name": "query-chardev-backends"}, {"name": "query-chardev"}, {"name": "query-events"}, {
"name": "query-commands"}, {"name": "query-version"}, {"name": "human-monitor-command"}, {"name": "qmp_capabilities"}, {"name": "add_client"}, {"n
ame": "expire_password"}, {"name": "set_password"}, {"name": "block_set_io_throttle"}, {"name": "block_passwd"}, {"name": "query-fdsets"}, {"name"
: "remove-fd"}, {"name": "add-fd"}, {"name": "closefd"}, {"name": "getfd"}, {"name": "set_link"}, {"name": "balloon"}, {"name": "drive-mirror"}, {
"name": "blockdev-snapshot-delete-internal-sync"}, {"name": "blockdev-snapshot-internal-sync"}, {"name": "blockdev-snapshot-sync"}, {"name": "tran
saction"}, {"name": "block-job-complete"}, {"name": "block-job-resume"}, {"name": "block-job-pause"}, {"name": "block-job-cancel"}, {"name": "bloc
k-job-set-speed"}, {"name": "drive-backup"}, {"name": "block-commit"}, {"name": "block-stream"}, {"name": "block_resize"}, {"name": "object-del"},
 {"name": "object-add"}, {"name": "netdev_del"}, {"name": "netdev_add"}, {"name": "query-dump-guest-memory-capability"}, {"name": "dump-guest-memo
ry"}, {"name": "client_migrate_info"}, {"name": "migrate_set_downtime"}, {"name": "migrate_set_speed"}, {"name": "query-migrate-cache-size"}, {"na
me": "migrate-set-cache-size"}, {"name": "migrate_cancel"}, {"name": "migrate"}, {"name": "xen-set-global-dirty-log"}, {"name": "xen-save-devices-
state"}, {"name": "ringbuf-read"}, {"name": "ringbuf-write"}, {"name": "inject-nmi"}, {"name": "pmemsave"}, {"name": "memsave"}, {"name": "cpu-add
"}, {"name": "cpu"}, {"name": "send-key"}, {"name": "device_del"}, {"name": "device_add"}, {"name": "system_powerdown"}, {"name": "system_reset"},
 {"name": "system_wakeup"}, {"name": "cont"}, {"name": "stop"}, {"name": "screendump"}, {"name": "change"}, {"name": "eject"}, {"name": "quit"}]}
{"execute": "query-block"}
{"return": [{"io-status": "ok", "device": "ide0-hd0", "locked": false, "removable": false, "inserted": {"iops_rd": 0, "image": {"virtual-size": 53
68709120, "filename": "ubuntutest.qcow2", "cluster-size": 65536, "format": "qcow2", "actual-size": 790396928, "format-specific": {"type": "qcow2",
 "data": {"compat": "1.1", "lazy-refcounts": false}}, "dirty-flag": false}, "iops_wr": 0, "ro": false, "backing_file_depth": 0, "drv": "qcow2", "i
ops": 0, "bps_wr": 0, "encrypted": false, "bps": 0, "bps_rd": 0, "file": "ubuntutest.qcow2", "encryption_key_missing": false}, "type": "unknown"},
 {"io-status": "ok", "device": "ide1-cd0", "locked": false, "removable": true, "tray_open": false, "type": "unknown"}, {"device": "floppy0", "lock
ed": false, "removable": true, "tray_open": false, "type": "unknown"}, {"device": "sd0", "locked": false, "removable": true, "tray_open": false, "
type": "unknown"}]}
```

2.11 QEMU-KVM: Monitoring and Debugging

- 访问The QEMU Monitor Protocol (QMP)：
  - 通过telnet
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -qmp tcp:localhost:4444,server

```
root@popsuper1982:/home/openstack/images# qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=vi
rtio -net tap,ifname=tap0,script=no,downscript=no -qmp tcp:localhost:4444,server
QEMU waiting for connection on: tcp:[::1]:4444,server
```

```
root@popsuper1982:/home/openstack/images# telnet localhost 4444
Trying ::1...
Connected to localhost.
Escape character is '^]'.
{"QMP": {"version": {"qemu": {"micro": 0, "minor": 0, "major": 2}, "package": " (Debian 2.0.0+dfsg-2ubuntu1.2)"}, "capabilities": []}}
{"execute": "qmp_capabilities"}
{"return": {}}
{"execute": "query-block"}
{"return": [{"io-status": "ok", "device": "ide0-hd0", "locked": false, "removable": false, "inserted": {"iops_rd": 0, "image": {"virtual-size": 53
68709120, "filename": "ubuntutest.qcow2", "cluster-size": 65536, "format": "qcow2", "actual-size": 790396928, "format-specific": {"type": "qcow2",
 "data": {"compat": "1.1", "lazy-refcounts": false}}, "dirty-flag": false}, "iops_wr": 0, "ro": false, "backing_file_depth": 0, "drv": "qcow2", "i
ops": 0, "bps_wr": 0, "encrypted": false, "bps": 0, "bps_rd": 0, "file": "ubuntutest.qcow2", "encryption_key_missing": false}, "type": "unknown"},
 {"io-status": "ok", "device": "ide1-cd0", "locked": false, "removable": true, "tray_open": false, "type": "unknown"}, {"device": "floppy0", "lock
ed": false, "removable": true, "tray_open": false, "type": "unknown"}, {"device": "sd0", "locked": false, "removable": true, "tray_open": false, "
type": "unknown"}]}
```

2.11 QEMU-KVM: Monitoring and Debugging

- 访问The QEMU Monitor Protocol (QMP)：
  - 通过unix sock
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -qmp unix:./qmp-sock,server
    - 使用qmp-shell
      - git clone git://git.qemu-project.org/qemu.git
      - 在qemu/scripts/qmp下面

```
root@popsuper1982:/home/openstack/images/qemu/scripts/qmp# ./qmp-shell ../../../qmp-sock
Welcome to the QMP low-level shell!
Connected to QEMU 2.0.0

(QEMU) query-
query-balloon                     query-cpus                          query-migrate-cache-size          query-tpm
query-block                       query-dump-guest-memory-capability  query-migrate-capabilities        query-tpm-models
query-block-jobs                  query-events                        query-name                        query-tpm-types
query-blockstats                  query-fdsets                        query-named-block-nodes           query-uuid
query-chardev                     query-iothreads                     query-pci                         query-version
query-chardev-backends            query-kvm                           query-rx-filter                   query-vnc
query-command-line-options        query-machines                      query-spice
query-commands                    query-mice                          query-status
query-cpu-definitions             query-migrate                       query-target
(QEMU) query-block
{u'return': [{u'locked': False, u'type': u'unknown', u'io-status': u'ok', u'removable': False, u'device': u'ide0-hd0', u'inserted': {u'bps_rd': 0,
 u'backing_file_depth': 0, u'encrypted': False, u'image': {u'cluster-size': 65536, u'format': u'qcow2', u'filename': u'ubuntutest.qcow2', u'virtua
l-size': 5368709120, u'dirty-flag': False, u'format-specific': {u'data': {u'compat': u'1.1', u'lazy-refcounts': False}, u'type': u'qcow2'}, u'actu
al-size': 790396928}, u'bps_wr': 0, u'drv': u'qcow2', u'bps': 0, u'iops': 0, u'file': u'ubuntutest.qcow2', u'iops_rd': 0, u'encryption_key_missing
': False, u'ro': False, u'iops_wr': 0}}, {u'locked': False, u'tray_open': False, u'io-status': u'ok', u'removable': True, u'device': u'ide1-cd0',
u'type': u'unknown'}, {u'device': u'floppy0', u'type': u'unknown', u'tray_open': False, u'locked': False, u'removable': True}, {u'device': u'sd0',
 u'type': u'unknown', u'tray_open': False, u'locked': False, u'removable': True}]}
(QEMU)
```

2.11 QEMU-KVM: Monitoring and Debugging

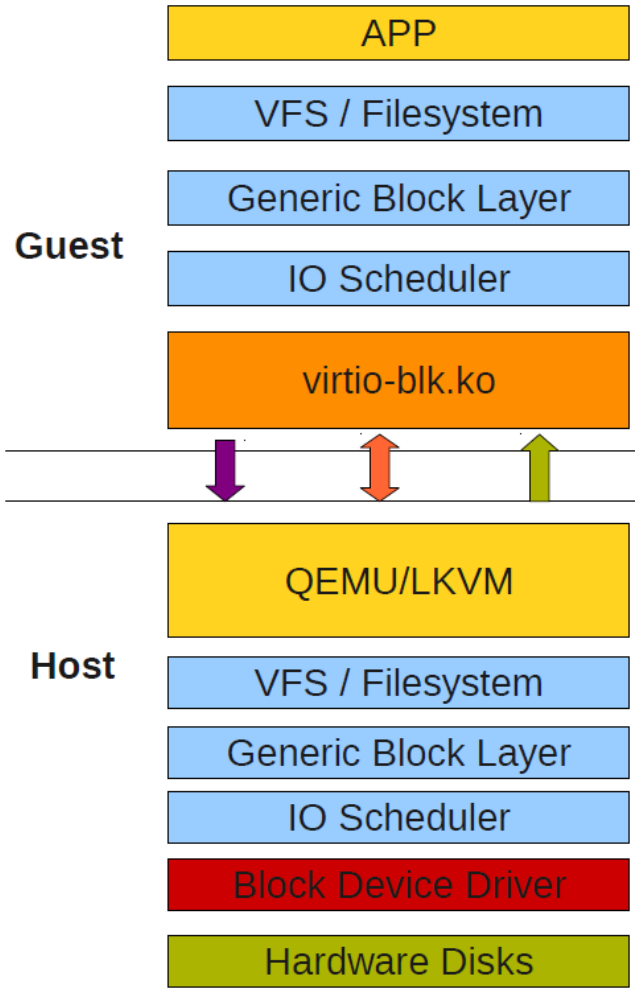
- GDB with Qemu
  - 监听gdb端口
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no --monitor stdio -gdb tcp::1234
  - Gdb进行连接
    - gdb
    - target remote localhost:1234

2.12 QEMU-KVM: KVM性能和最佳实践

- 最佳实践一：使用半虚拟化驱动virtio
  - 低延迟，高吞度量
  - 兼容性差
  - 对于需要高吞吐量的设备，往往硬件方面会有特殊的实现，而模拟设备无法利用
  - 对于block storage，使用virtio_blk
  - 对于network，使用virtio_net
  - 对于内存，使用virtio_balloon，提供memory over-commitment

2.12 QEMU-KVM: KVM性能和最佳实践

- 最佳实践二：VM的存储设备
  - 尽量使用block device
    - 性能更好
    - 无需管理HOST文件系统
    - 无需管理稀疏文件
    - I/O Cache以4K为边界
  - 如果不能使用block device，则只好使用Image File
    - 易管理，易移动
    - 需要HOST文件系统，需要管理稀疏文件
    - Partition问题：
      - 标准分区软件分区是以512byte一个Cylinder为边界的
      - Linux系统的I/O Cache是以4K为边界的
      - 所以Linux下的标准文件系统将分区format的时候，会将文件以4k为边界进行存储，从而每次缓存正好缓存整个page(大小4k)
      - Image File作为一个文件，在Host文件系统中一定是以4k为边界的
      - 然而一个Image File对于Guest来讲是整块硬盘，对其分区的时候往往不是以4K为边界的
      - 从而Guest上读取文件的时候，本来是读取缓存的4K一个page，映射到Host文件系统上，变成了两个Page。写入也要更新两个页。



2.12 QEMU-KVM: KVM性能和最佳实践

- 最佳实践三：CPU超配
  - 每个Guest相当于一个进程，Guest中的每个vcpu相当于一个线程
  - Host CPU支持进程间切换，SMP可以并行执行多个进程，从而可以进行CPU超配
  - CPU的超配增加了进程上下文切换，从而可能带来性能问题
  - 为了保证Guest进行能够得到足够的时间片，常常使用cgroup的cpu.cfs_period_us和cpu.cfs_quota_us来控制
  - Cfs全称Completely Fair Scheduler是Linux Kernel的调度策略
  - cfs_period_us的意思是cgroup对CPU的调度的干预周期，cfs_quota_us是指则一个周期内这个进程得到的时间片长度。比如cfs_period_us=100000说明100毫秒cgroup进行一次干预，cfs_quota_us=25000表示在100毫秒里面，这个进程能够得到25毫秒的时间片，如果对cgroup进行的修改，则要等到下个100毫秒才起作用
  - Processor Pin: 可以讲vCPU pin到一个或者一组共享cache的物理CPU上，由于cache共享，则很多命令和数据都被缓存，从而提高性能。缺点是即便其他物理CPU空闲，由于绑定到了这个CPU，也得不到运行

2.12 QEMU-KVM: KVM性能和最佳实践

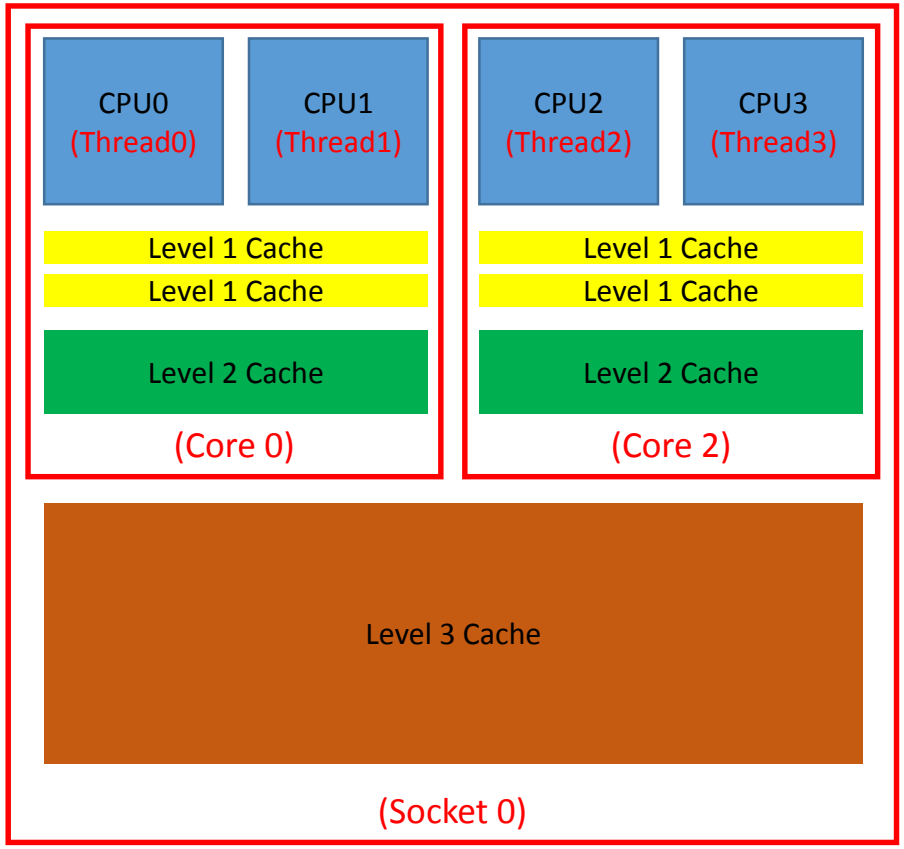
- 最佳实践三：CPU超配
  - CPU和Cache拓扑模型/sys/devices/system/cpu
  - virsh vcpupin guest# vproc# pproc#,pproc#

有四个CPU
ls /sys/devices/system/cpu
cpu0 cpu1 cpu2 cpu3

cpu0/topology/core_id = 0
cpu1/topology/core_id = 0
cpu2/topology/core_id = 2
cpu3/topology/core_id = 2

cpu0/topology/core_siblings_list = 0-3
cpu0/topology/thread_siblings_list = 0-1
cpu2/topology/thread_siblings_list = 2-3

cpu0/cache/index0/level = 1
cpu0/cache/index0/shared_cpu_list = 0-1
cpu0/cache/index1/level = 1
cpu0/cache/index1/shared_cpu_list = 0-1
cpu0/cache/index2/level = 2
cpu0/cache/index2/shared_cpu_list = 0-1
cpu0/cache/index3/level = 3
cpu0/cache/index3/shared_cpu_list = 0-3

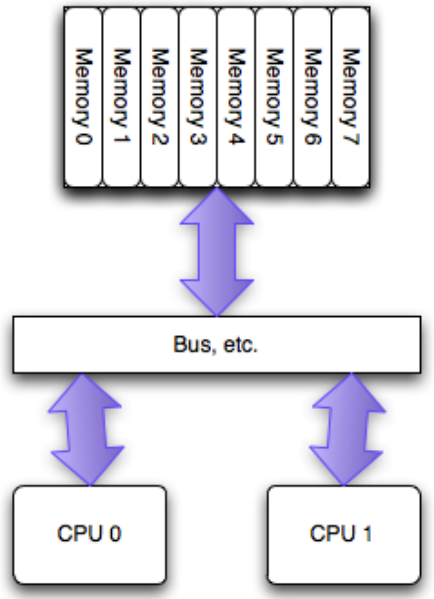


2.12 QEMU-KVM: KVM性能和最佳实践

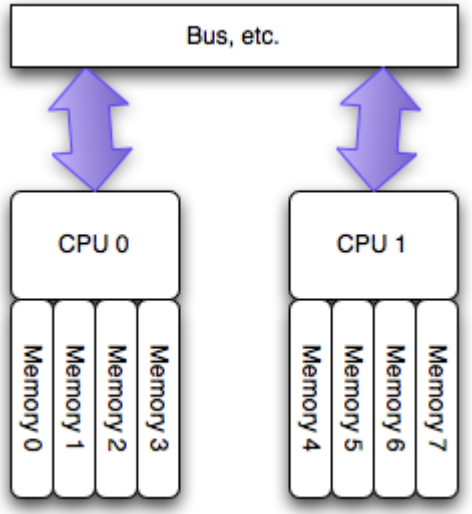
- 最佳实践四：内存超配
  - KSM全称Kernel Same Page Merging
  - Qemu向KSM注册内存，KSM对内存进行扫描，将相同的内存区域设为共享，并且copy on write
  - # cat /boot/config-3.13.0-27-generic | grep KSM
    CONFIG_KSM=y
  - 共享内存节约内存空间，但是内存扫描同时影响性能
  - 使用virtio_balloon
  - 尽量不要使用swap，/proc/sys/vm/swappiness设为0

2.12 QEMU-KVM: KVM性能和最佳实践

- 最佳实践四：内存超配
  - 多核的两种方式：SMP和NUMA
    - SMP的问题主要在CPU和内存之间的通信延迟较大、通信带宽受限于系统总线带宽，同时总线带宽会成为整个系统的瓶颈
    - NUMA（Non-Uniform Memory Access）：每个处理器有其可以直接访问其自身的“本地”内存池，使CPU和这块儿内存之间拥有更小的延迟和更大的带宽。而且整个内存仍然可做为一个整体，可以接受来自任何CPU的访问。
  - 禁止zone_reclaim_mode：/proc/sys/vm/zone_reclaim_mode
    - 当操作系统分配内存时，发现CPU本节点的内存已满，如果reclaim=true，则将会回收一部分本地内存，否则将分配其他节点的内存
    - 使用KSM和Huge Page都会造成reclaim



SMP



NUMA

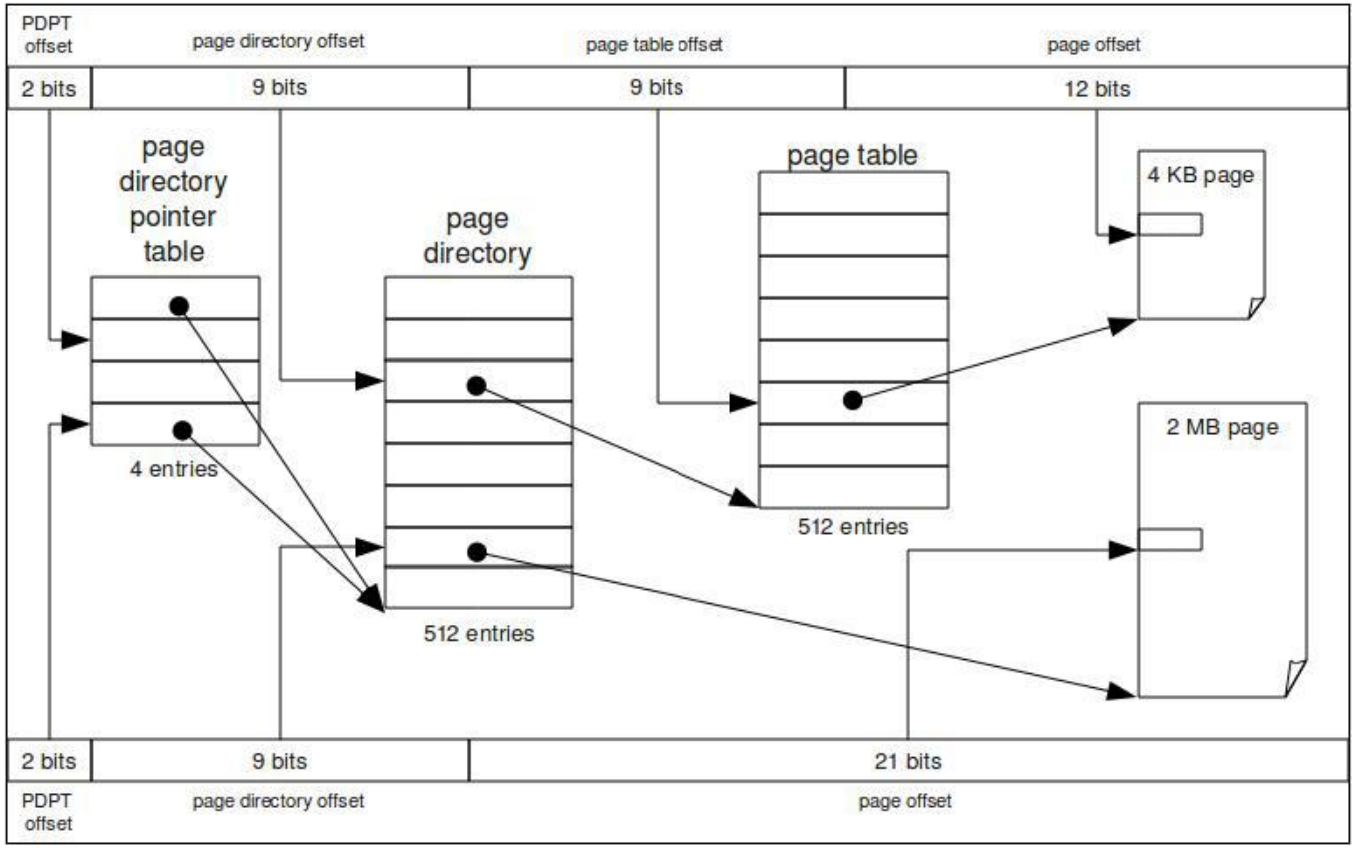
2.12 QEMU-KVM: KVM性能和最佳实践

- 最佳实践四：内存超配
  - Huge pages

Translation Lookaside Buffer (TLB)是一个从虚拟地址到物理地址转换的一个Cache

Huge Page不但使得每次地址转换比较快，而且使得TLB每个记录较小，相同的内存的情况下，保存的记录较多，Cache命中率较高

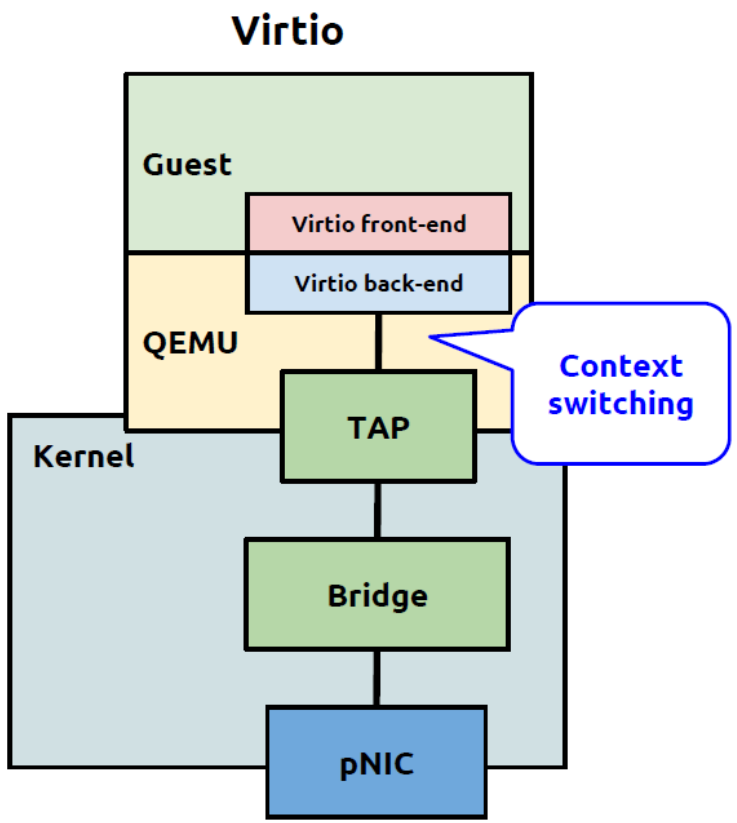
普通页：PDPT -> PD -> Page Table -> In Page Offset

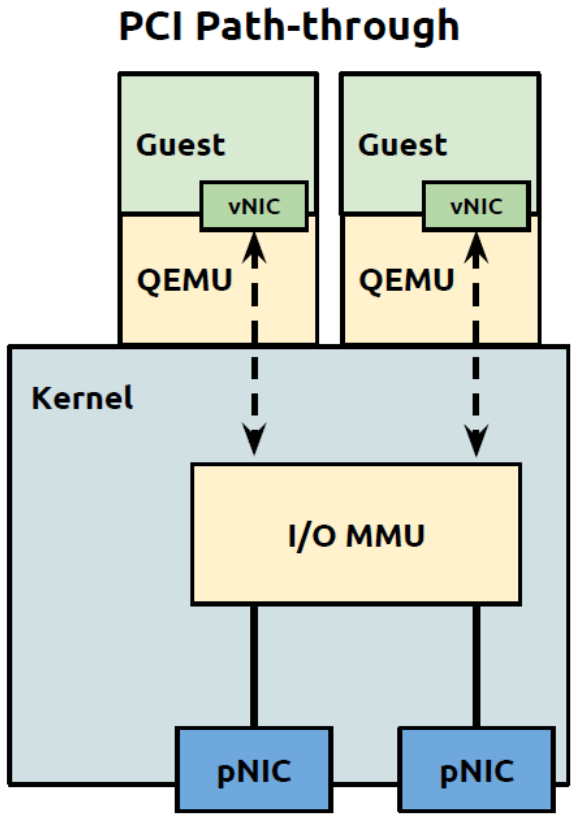


大页：PDPT -> PD -> In Page Offset

2.12 QEMU-KVM: KVM性能和最佳实践

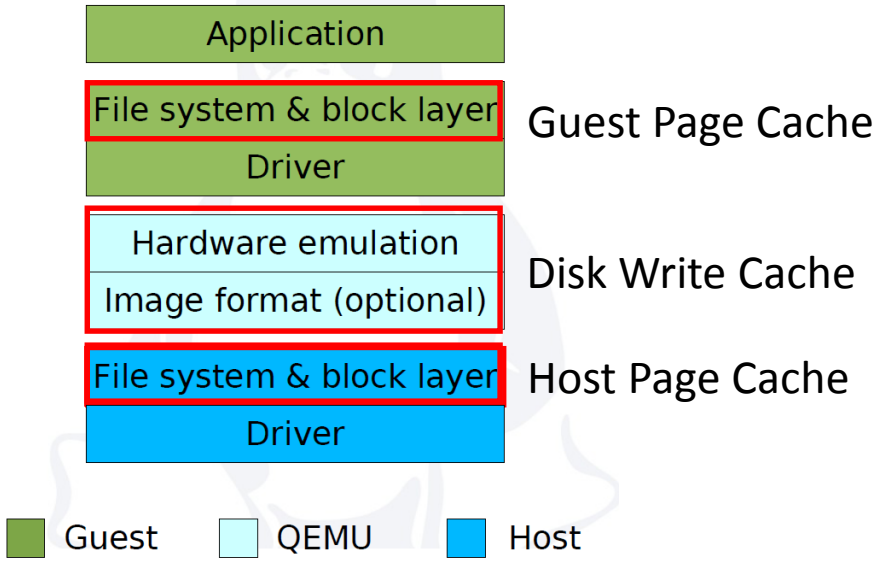
- 最佳实践五：网络
  - 使用tap作为网卡配置
  - 使用virtio_net
  - 使用PCI passthough可以提高性能，但是影响虚拟机迁移





2.12 QEMU-KVM: KVM性能和最佳实践

- 最佳实践六： Block I/O performance
  - Host使用ext3文件系统，因为ext4的barrier会影响性能
  - 使用raw image，比qcow2性能要好
  - 选择正确的cache策略
    - Page Cache: Host和Guest在FS和Block层都有自己的Page Cache，我们无法控制Guest里面的设置，但是可以配置Host Page Cache
    - Disk Write Cache在Hypervisor层
    - Cache Mode有以下四种，推荐使用none
    - I/O Scheduler推荐使用Deadline I/O scheduler



Mode	Host page cache	Guest disk write cache
none	off	on
writethrough	on	off
writeback	on	on
unsafe	on	ignored

Libvirt

接下来的章节介绍Libvirt工具

3.1 Libvirt: 管理Domain

- 将命令行转换为XML

qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -balloon virtio -drive file=ubuntutest.qcow2,if=virtio -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio

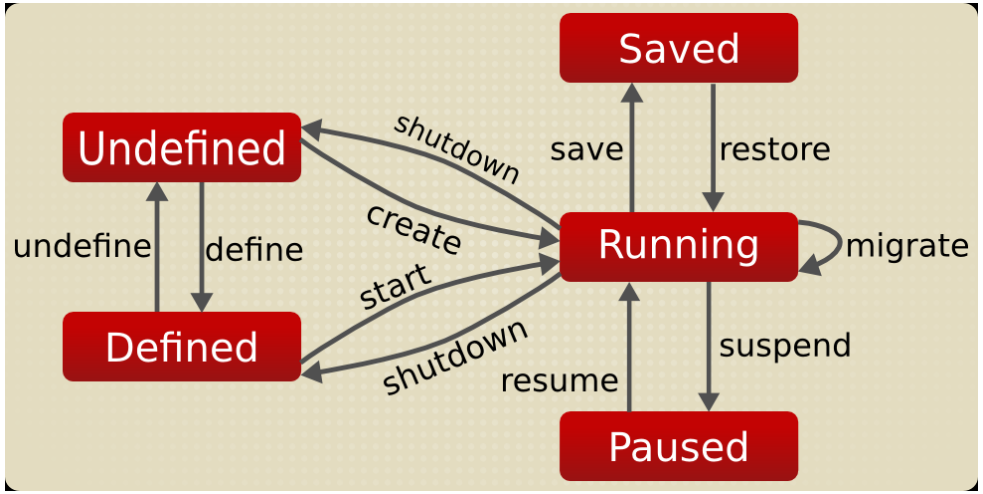
ubuntutest.xml

```
<domain type='kvm'>
  <name>ubuntutest</name>
  <uuid>0f0806ab-531d-6134-5def-c5b4955292aa</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/home/openstack/images/ubuntutest.qcow2'/>
      <target dev='vda' bus='virtio'/>
    </disk>
    <controller type='pci' index='0' model='pci-root'/>
    <interface type='bridge'>
      <mac address='fa:16:3e:6e:89:ce'/>
      <source bridge='br0'/>
      <target dev='tap1'/>
      <model type='virtio'/>
      <alias name='net0'/>
    </interface>
    <serial type='pty'>
      <target port='0'/>
    </serial>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'>
      <listen type='address' address='0.0.0.0'/>
    </graphics>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
      <alias name='video0'/>
    </video>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
    </memballoon>
  </devices>
</domain>
```

3.1 Libvirt: 管理Domain

- 定义Domain
  - virsh define ubuntutest.xml
- 编辑Domain
  - virsh edit ubuntutest
- 启动Domain
  - virsh start ubuntutest
- 关闭Domain
  - virsh shutdown ubuntutest
  - virsh destroy ubuntutest
- 重启Domain
  - virsh reboot ubuntutest
- 暂停Domain
  - virsh suspend ubuntutest
- 唤醒Domain
  - virsh resume ubuntutest
- 删除Domain
  - virsh undefined ubuntutest

```
/usr/bin/qemu-system-x86_64 -name ubuntutest -S -machine pc-i440fx-
trusty,accel=kvm,usb=off -m 1024 -realtime mlock=off -smp 1,sockets=1,cores=1,threads=1 -
uuid 0f0806ab-531d-6134-5def-c5b4955292aa -no-user-config -nodefaults -chardev
socket,id=charmonitor,path=/var/lib/libvirt/qemu/ubuntutest.monitor,server,nowait -mon
chardev=charmonitor,id=monitor,mode=control -rtc base=utc -no-shutdown -boot strict=on -
device piix3-usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2 -drive
file=/home/openstack/images/ubuntutest.qcow2,if=none,id=drive-virtio-disk0,format=qcow2
-device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x4,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=fa:16:3e:6e:89:ce,bus=pci.0,addr=0x3 -chardev
pty,id=charserial0 -device isa-serial,chardev=charserial0,id=serial0 -vnc 0.0.0.0:0 -device
cirrus-vga,id=video0,bus=pci.0,addr=0x2 -device virtio-balloon-
pci,id=balloon0,bus=pci.0,addr=0x5
```



3.2 Libvirt: 管理snapshot

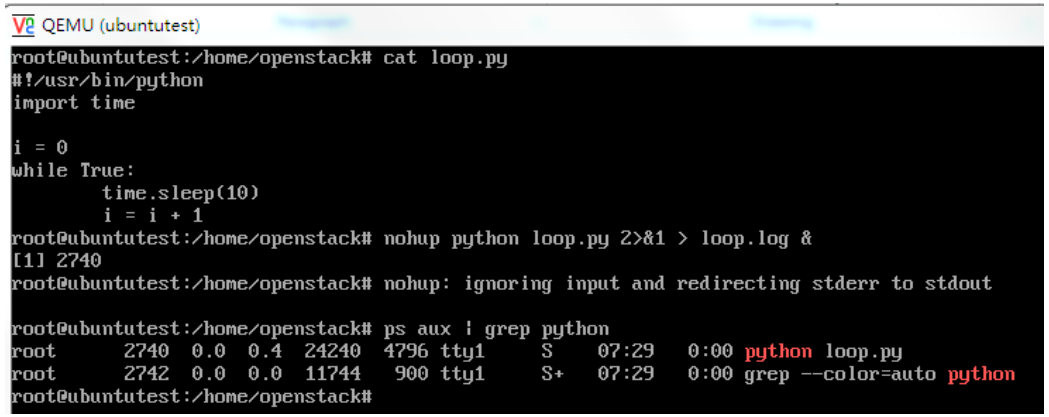
- Snapshot广义来讲分为三个级别
  - Volume Manager级别：常见的是LVM的snapshot，在openstack中，对block storage进行snapshot对应这个级别
    - lvcreate --size 100M --snapshot --name snap /dev/vg00/lvol1
  - 文件系统级别： OCFS2，常用的ext3不支持
  - 文件级别：raw文件不支持snapshot，qcow2支持snapshot，分两种
    - Internal Snapshot: snapshot保存在qcow2文件的内部
      - VM State snapshot: snapshot整个VM，而不仅仅是disk
      - Disk State snapshot: 仅仅snapshot这个disk
    - External Snapshot: 原来的qcow2成为read-only的模式，新的改变保存到另外的qcow2文件

3.2 Libvirt: 管理snapshot

- Internal Snapshot - VM State snapshot
  - 启动一个虚拟机

```
root@popsuper1982:/home/openstack/images# virsh list
 Id    Name                           State
----------------------------------------------------
 12    ubuntutest                     running
```

  - 在虚拟机里面启动一个python进程，里面无限循环



  - 将VM State保存在一个文件里面
    - virsh save ubuntutest ubuntutest_vmstate
    - 完毕后虚拟机被关闭

```
root@popsuper1982:/home/openstack/images# virsh save ubuntutest ubuntutest_vmstate

Domain ubuntutest saved to ubuntutest_vmstate

root@popsuper1982:/home/openstack/images# virsh list --all | grep ubuntutest
 -     ubuntutest                     shut off
root@popsuper1982:/home/openstack/images# ls -l ubuntutest_vmstate
-rw------- 1 root root 753069767 Oct  8 07:36 ubuntutest_vmstate
```

3.2 Libvirt: 管理snapshot

- Internal Snapshot - VM State snapshot
  - 直接启动虚拟机
    - virsh start ubuntutest
    - 启动一个全新的虚拟机，python进程不存在
  - 恢复虚拟机
    - virsh restore ubuntutest_vmstate
    - 虚拟机启动了

```
root@popsuper1982:/home/openstack/images# virsh restore ubuntutest_vmstate
Domain restored from ubuntutest_vmstate

root@popsuper1982:/home/openstack/images# virsh list --all | grep ubuntutest
 13    ubuntutest                     running
```

    - VNC登录虚拟机，发现python进行和原来的状态一样

3.2 Libvirt: 管理snapshot

- Internal Snapshot - Disk State snapshot
  - 虚拟机在运行状态下进行snapshot
  - 查看snapshot的信息，为空
    - virsh snapshot-list ubuntutest
  - 创建snapshot
    - virsh snapshot-create ubuntutest
    - 创建过程中虚拟机处于pause的状态

```
root@popsuper1982:/home/openstack/images# virsh snapshot-create ubuntutest
Domain snapshot 1412782509 created
```

  - 查看snapshot的信息

```
root@popsuper1982:/home/openstack/images# virsh snapshot-list ubuntutest
 Name                 Creation Time             State
------------------------------------------------------------
 1412782509           2014-10-08 08:35:09 -0700 running
```

  - 查看Image的信息

```
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest.qcow2
image: ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 1.6G
cluster_size: 65536
Snapshot list:
ID        TAG                 VM SIZE                DATE       VM CLOCK
1         1412782509             718M 2014-10-08 08:35:09  166:36:14.727
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

3.2 Libvirt: 管理snapshot

- Internal Snapshot - Disk State snapshot
  - 创建了internal snapshot后，vm无法undefined
    - virsh destroy ubuntutest

```
root@popsuper1982:/home/openstack/images# virsh undefine ubuntutest
error: Failed to undefine domain ubuntutest
error: Requested operation is not valid: cannot delete inactive domain with 1 snapshots

root@popsuper1982:/home/openstack/images# virsh snapshot-list ubuntutest
 Name                 Creation Time             State
------------------------------------------------------------
 1412782509           2014-10-08 08:35:09 -0700 running
```

  - 删除snapshot
    - virsh snapshot-delete ubuntutest 1412782509

```
root@popsuper1982:/home/openstack/images# virsh snapshot-delete ubuntutest 1412782509
Domain snapshot 1412782509 deleted

root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest.qcow2
image: ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 971M
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

3.2 Libvirt: 管理snapshot

- External Snapshot
  - 查看block storage
    - virsh domblklist ubuntutest –details

```
root@popsuper1982:/home/openstack/images# virsh domblklist ubuntutest --details
Type       Device     Target     Source
------------------------------------------------
file       disk       vda        /home/openstack/images/ubuntutest.qcow2
```

  - 创建一个snapshot
    - virsh snapshot-create-as ubuntutest snap1-ubuntutest "snap1 description" --diskspec vda,file=/home/openstack/snap1-ubuntutest.qcow2 --disk-only --atomic
    - 如果报下面的错误，则是apparmor在作怪，参考下面的操作：

```
root@popsuper1982:/home/openstack/images# virsh snapshot-create-as ubuntutest snap1-ubuntutest "snap1 description" --diskspec vda,file=/home/opens
tack/snap1-ubuntutest.qcow2 --disk-only --atomic
error: internal error: unable to execute QEMU command 'transaction': Could not open '/home/openstack/images/ubuntutest.qcow2': Could not open '/ho
me/openstack/images/ubuntutest.qcow2': Permission denied: Permission denied
```

      - 修改/etc/libvirt/qemu.conf，其中security_driver = [“selinux”, “apparmor”]改为security_driver = "none“
      - 重启libvirt: service libvirt-bin restart
      - 重启虚拟机：virsh destroy ubuntutest，virsh start ubuntutest
      - 可以正确执行了

```
root@popsuper1982:/home/openstack/images# virsh snapshot-create-as ubuntutest snap1-ubuntutest "snap1 description" --diskspec vda,file=/home/opens
tack/snap1-ubuntutest.qcow2 --disk-only --atomic
Domain snapshot snap1-ubuntutest created
```

3.2 Libvirt: 管理snapshot

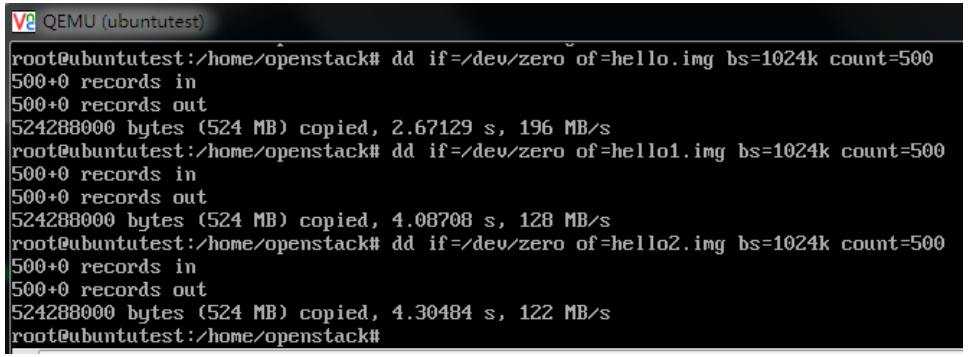
- External Snapshot
  - 虚拟机使用新的snapshot作为硬盘，在snapshot后创建一个512M的文件

```
root@popsuper1982:/home/openstack/images# qemu-img info /home/openstack/snap1-ubuntutest.qcow2
image: /home/openstack/snap1-ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 586M
cluster_size: 65536
backing file: /home/openstack/images/ubuntutest.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
root@popsuper1982:/home/openstack/images# virsh domblklist ubuntutest
Target     Source
------------------------------------------------
vda        /home/openstack/snap1-ubuntutest.qcow2
```

  - 再创建两个snapshot，每打一次snapshot都创建一个512M的文件
    - virsh snapshot-create-as ubuntutest snap2-ubuntutest "snap2 description" --diskspec vda,file=/home/openstack/snap2-ubuntutest.qcow2 --disk-only --atomic
    - virsh snapshot-create-as ubuntutest snap3-ubuntutest "snap3 description" --diskspec vda,file=/home/openstack/snap3-ubuntutest.qcow2 --disk-only --atomic

3.2 Libvirt: 管理snapshot

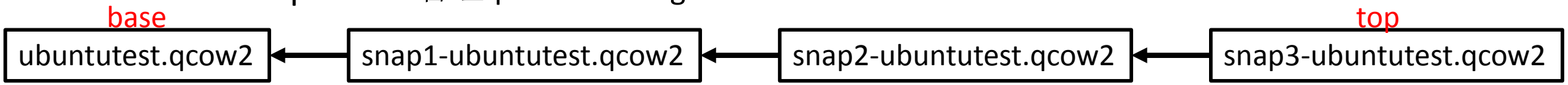
- External Snapshot
  - 虚拟机使用新的snapshot作为硬盘
  - 查看文件backing_file链
  - 查看snapshot



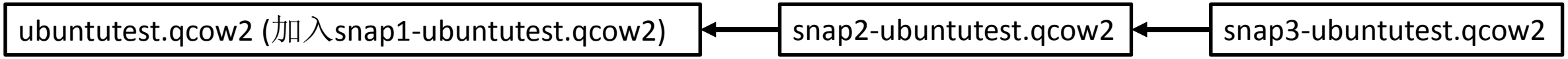

```
root@popsuper1982:/home/openstack/images# virsh domblklist ubuntutest
Target     Source
------------------------------------------------
vda        /home/openstack/snap3-ubuntutest.qcow2

root@popsuper1982:/home/openstack/images# qemu-img info /home/openstack/snap2-ubuntutest.qcow2
image: /home/openstack/snap2-ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 501M
cluster_size: 65536
backing file: /home/openstack/snap1-ubuntutest.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
root@popsuper1982:/home/openstack/images# qemu-img info /home/openstack/snap3-ubuntutest.qcow2
image: /home/openstack/snap3-ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 501M
cluster_size: 65536
backing file: /home/openstack/snap2-ubuntutest.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
root@popsuper1982:/home/openstack/images# virsh snapshot-list ubuntutest --tree
snap1-ubuntutest
  |
  +- snap2-ubuntutest
     |
     +- snap3-ubuntutest
```

3.2 Libvirt: 管理snapshot

- External Snapshot： 管理qcow2 backing chain

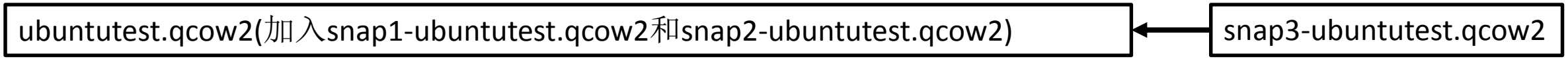


  - 方式一： virsh blockcommit/virDomainBlockCommit
    - 将内容从snapshot commit到base
    - virsh blockcommit ubuntutest vda --base /home/openstack/images/ubuntutest.qcow2 --top /home/openstack/snap1-ubuntutest.qcow2 --wait --verbose
    - virDomainBlockCommit(dom, "vda", "ubuntutest.qcow2", "snap1-ubuntutest.qcow2", 0, 0)



    - virDomainBlockCommit(dom, "vda", "snap1-ubuntutest.qcow2 ", "snap2-ubuntutest.qcow2", 0, 0)

    - virDomainBlockCommit(dom, "vda", " ubuntutest.qcow2", "snap2-ubuntutest.qcow2", 0, 0)



    - 最顶层的和base的都不能被commit，因而commit之后的结果最少有两层，一层base，一层top

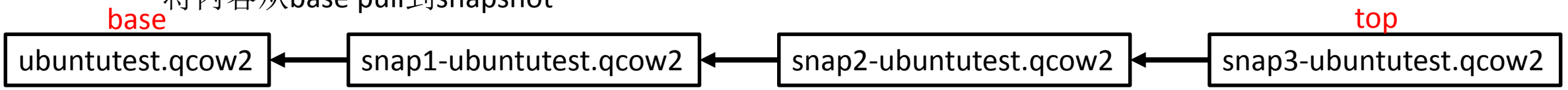
3.2 Libvirt: 管理snapshot

- External Snapshot: Blockcommit
  - virsh blockcommit ubuntutest vda --base /home/openstack/images/ubuntutest.qcow2 --top /home/openstack/snap2-ubuntutest.qcow2 --wait --verbose
  - Snap2和snap1的512M都合并到了ubuntutest.qcow2中

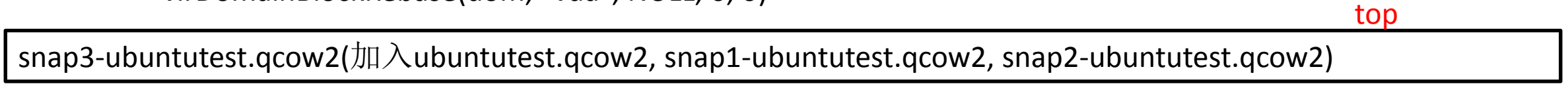
```
root@popsuper1982:/home/openstack/images# qemu-img info /home/openstack/snap3-ubuntutest.qcow2
image: /home/openstack/snap3-ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 501M
cluster_size: 65536
backing file: /home/openstack/images/ubuntutest.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
root@popsuper1982:/home/openstack/images# qemu-img info /home/openstack/images/ubuntutest.qcow2
image: /home/openstack/images/ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 1.7G
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

3.2 Libvirt: 管理snapshot

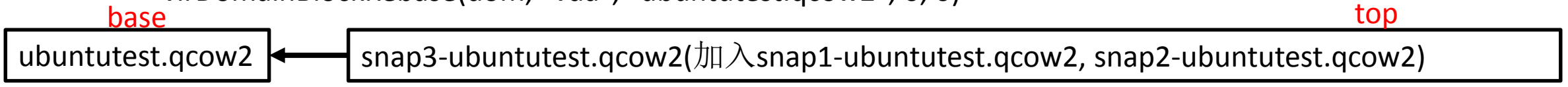
- External Snapshot： 管理qcow2 backing chain
  - 方式二： virsh blockpull/virDomainBlockRebase
  - 将内容从base pull到snapshot



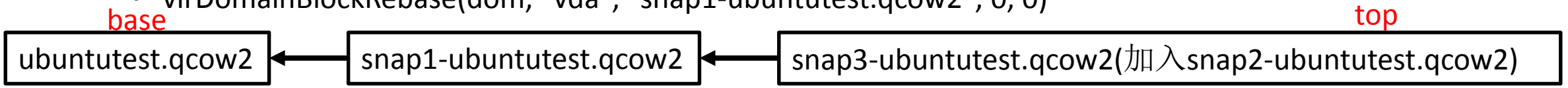
  - virsh blockpull ubuntutest --path /home/openstack/snap3-ubuntutest.qcow2 --base /home/openstack/images/ubuntutest.qcow2 --wait –verbose
  - virDomainBlockRebase(dom, "vda", NULL, 0, 0)



  - virDomainBlockRebase(dom, "vda", "ubuntutest.qcow2", 0, 0)



  - virDomainBlockRebase(dom, "vda", "snap1-ubuntutest.qcow2", 0, 0)



  - 仅仅能够pull到top的image，而不能是其他的snapshot

3.2 Libvirt: 管理snapshot

- External Snapshot: Blockpoll
  - virsh blockpull ubuntutest --path /home/openstack/snap3-ubuntutest.qcow2 --wait --verbose
  - Ubuntutest.qcow2的内容并入snap3-ubuntutest.qcow2，snap3独立存在，是虚拟机的当前active硬盘

```
root@popsuper1982:/home/openstack/images# qemu-img info /home/openstack/snap3-ubuntutest.qcow2
image: /home/openstack/snap3-ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 502M
cluster_size: 65536
backing file: /home/openstack/images/ubuntutest.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
root@popsuper1982:/home/openstack/images# virsh blockpull ubuntutest  --path /home/openstack/snap3-ubuntutest.qcow2 --wait --verbose
Block Pull: [100 %]
Pull complete
root@popsuper1982:/home/openstack/images# qemu-img info /home/openstack/snap3-ubuntutest.qcow2
image: /home/openstack/snap3-ubuntutest.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 1.9G
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
root@popsuper1982:/home/openstack/images# virsh domblklist ubuntutest
Target     Source
------------------------------------------------
vda        /home/openstack/snap3-ubuntutest.qcow2
```

3.2 Libvirt: 管理snapshot

- External Snapshot
  - Libvirt对backing_file链和snapshot列表是分开管理的，backing_file链的改变并不会改变snapshot-list
  - 这个时候virsh snapshot-list ubuntutest –tree还是原来的结果
  - 如果在改变backing_file链之后，要删除snapshot-list，可以只删除metadata
    - virsh snapshot-delete --metadata ubuntutest snap1-ubuntutest
  - 当然也可以在创建snapshot的时候，不创建metadata
    - virsh snapshot-create --no-metadata

3.2 Libvirt: 管理snapshot

- External Snapshot: 方式三：blockcopy，将内容复制到另一个Image
  - Openstack中snapshot就是这样实现的
  - 通常有一个base Image: /home/openstack/images/ubuntutest.qcow2
  - 为每一个虚拟机创建一个单独的硬盘
    - qemu-img create -f qcow2 -o backing_file=/home/openstack/images/ubuntutest.qcow2 ubuntutest_origin.qcow2
  - 创建transient domain的xml
    - ubuntutest_transient.xml

```
<domain type='kvm' id='23'>
  <name>ubuntutest_transient</name>
  <uuid>0f0806ab-531d-6134-5def-c5b4955292bb</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/home/openstack/images/ubuntutest_origin.qcow2'/>
      <target dev='vda' bus='virtio'/>
      <alias name='virtio-disk0'/>
    </disk>
```

3.2 Libvirt: 管理snapshot

- External Snapshot: 方式三： blockcopy，将内容复制到另一个Image
  - 创建一个transient的domain
    - virsh create ubuntutest_transient.xml

```
root@popsuper1982:/home/openstack/images# virsh create ubuntutest_transient.xml
Domain ubuntutest_transient created from ubuntutest_transient.xml

root@popsuper1982:/home/openstack/images# virsh list
 Id    Name                           State
----------------------------------------------------
 25    ubuntutest_transient           running
```

  - 在虚拟机里面创建512M的文件，dd if=/dev/zero of=hello.img bs=1024k count=500

```
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest_origin.qcow2
image: ubuntutest_origin.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 505M
cluster_size: 65536
backing file: /home/openstack/images/ubuntutest.qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

  - 创建一个空的snapshot文件，指向base Image
    - qemu-img create -f qcow2 -o backing_file=/home/openstack/images/ubuntutest.qcow2 ubuntutest_snapshot.qcow2

3.2 Libvirt: 管理snapshot

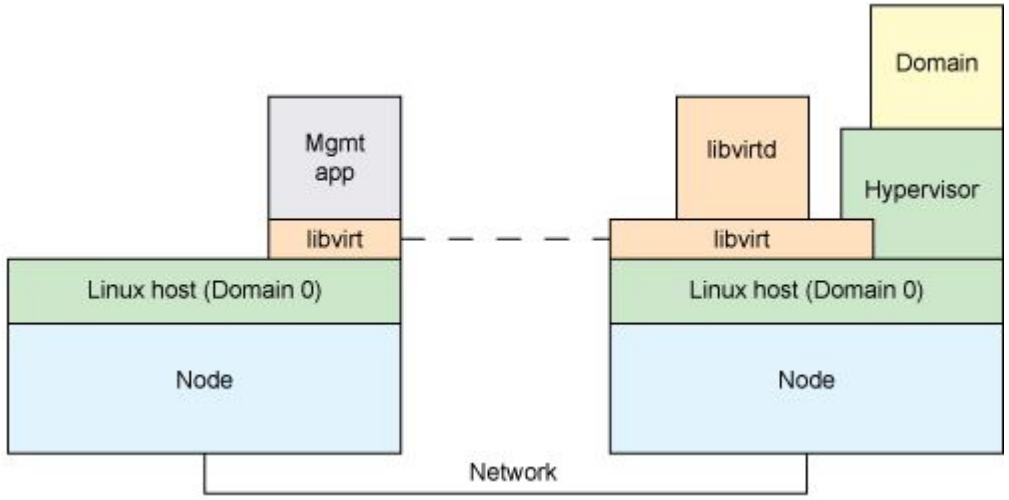
- External Snapshot: 方式三： blockcopy，将内容复制到另一个Image
  - 进行block copy
    - virsh blockcopy --domain ubuntutest_transient --path /home/openstack/images/ubuntutest_origin.qcow2 --dest /home/openstack/images/ubuntutest_snapshot.qcow2 --shallow --reuse-external --wait –verbose

```
root@popsuper1982:/home/openstack/images# virsh blockcopy --domain ubuntutest_transient --path /home/openstack/images/ubuntutest_origin.qcow2 --dest /home/openstack/images/
ubuntutest_snapshot.qcow2 --shallow --reuse-external --wait --verbose
Block Copy: [100 %]
Now in mirroring phase
root@popsuper1982:/home/openstack/images# qemu-img info ubuntutest_snapshot.qcow2
image: ubuntutest_snapshot.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 505M
cluster_size: 65536
backing file: /home/openstack/images/ubuntutest.qcow2
Format specific information:
    compat: 1.1
```

    - 从ubuntu_snapshot.qcow2启动查看，hello.img存在
    - 这里必须创建transient domain，否则会报如下错误
      - error: Requested operation is not valid: domain is not transient

3.3 Libvirt: Remote Access

- Libvirt提供远程管理虚拟机功能
- 三个基本问题
  - 使用什么URI进行远程连接
  - Authentication，如何验证你是谁
  - Authorization/Access Control，验证你有没有权限



3.3 Libvirt: Remote Access

- 使用什么URI进行远程连接
  - 格式：
    - driver[+transport]://[username@][hostname][:port]/[path][?extraparameters]
  - Transport有下面几种方式：
    - Unix：
      - Unix domain socket，仅仅在本地能够使用，不加密，使用unix本身的权限控制
      - qemu+unix:///system?socket=/opt/libvirt/run/libvirt/libvirt-sock
    - Ssh：
      - 通过ssh tunnel进行远程连接，相当于通过ssh tunnel到目标机器上调用unix domain socket，加密，使用unix本身的权限控制，能ssh通过username/password登录hostname，就能执行命令
      - qemu+ssh://root@example.com/system
    - TCP：
      - 通过TCP进行远程连接，通过DIGEST-MD5进行加密，使用SASL/Kerberos进行权限控制
      - qemu+tcp://example.com/system
    - TLS:
      - 和TCP类似，但是使用SSL对TCP进行加密，因而需要配置Key和Certificate，最好disable DIGEST-MD5避免加密两次，同样使用SASL/Kerberos进行权限控制
      - qemu+tls://root@mercury.example.com/system

3.3 Libvirt: Remote Access

- Unix
  - 使用root用户，可以virsh list
  - 使用openstack用户，则不可以
    - error: Failed to connect socket to '/var/run/libvirt/libvirt-sock': Permission denied
  - 在/etc/libvirt/libvirtd.conf中有下面的配置项：
    - unix_sock_group = "libvirtd"
    - unix_sock_ro_perms = "0777"
    - unix_sock_rw_perms = "0770"
  - 仅仅用户属于group “libvirtd”的时候，才能够访问sock
    - usermod -G libvirtd -a openstack
- Ssh
  - 使用SSH Tunnel，其实也是登录到另一台机器上连接sock
  - 如果上面的权限设置正确，则ssh也能成功
  - virsh -c qemu+ssh://openstack@192.168.57.1/system list

```
root@ubuntutest:/home/openstack# virsh -c qemu+ssh://openstack@192.168.57.1/system list
The authenticity of host '192.168.57.1 (192.168.57.1)' can't be established.
ECDSA key fingerprint is 6f:3d:0f:f6:e8:e6:28:44:4b:77:0b:d3:26:c3:a4:ce.
Are you sure you want to continue connecting (yes/no)? yes
openstack@192.168.57.1's password:
 Id    Name                           State
----------------------------------------------------
 26    ubuntutest                     running
```

3.3 Libvirt: Remote Access

- TCP
  - 在/etc/libvirt/libvirtd.conf中有下面的配置项：
    - listen_tcp = 1
    - tcp_port = "16509"
    - auth_tcp = "none“
  - 修改/etc/default/libvirt-bin
    - libvirtd_opts="-d -l"
  - 重启libvirt
    - 在/var/log/libvirt/libvirtd.log中报下面的错误
      - error : virNetTLSContextCheckCertFile:117 : Cannot read CA certificate '/etc/pki/CA/cacert.pem': No such file or directory

3.3 Libvirt: Remote Access

/etc/hosts里面添加
192.168.57.1 popsuper1982
192.168.57.100 ubuntutest

- 创建CA的key和Certificate
  - 安装certtool
    - apt-get install gnutls-bin
  - 假设自己是一个CA，而且是一个root CA
  - 创建一个template，声明自己是CA

```
# cat certificate_authority_template.info
cn = libvirt.org
ca
cert_signing_key
```

  - 生成一个CA的private key
    - umask 277 && certtool --generate-privkey > certificate_authority_key.pem
- CA有一个certificate，里面放着CA的public key，要生成这个certificate，则需要写一个certificate request，这个证书请求需要更高级的CA用它的private key对这个证书请求进行签发，方才能被大家公认(如果大家相信的是更高级的CA，则可以用更高级的CA的certificate审核这个证书)
- 由于这里的CA是root CA，没有更高级的CA了，所以要进行自签发，用自己的private key对自己的certificate请求进行签发
  - certtool --generate-self-signed --template certificate_authority_template.info --load-privkey certificate_authority_key.pem --outfile certificate_authority_certificate.pem
- 将证书拷贝到相应位置
  - cp certificate_authority_certificate.pem /etc/pki/CA/cacert.pem
  - chmod 444 /etc/pki/CA/cacert.pem

3.3 Libvirt: Remote Access

- 创建popsuper1982这台机器的key和certificate
  - popsuper1982是一个普通的机构
  - 生成一个template

```
# cat popsuper1982_server_template.info
organization = libvirt.org
cn = popsuper1982
tls_www_server
encryption_key
signing_key
```

  - 这个普通的机构需要有自己的private key
    - umask 277 && certtool --generate-privkey > popsuper1982_server_key.pem
  - 也需要一个证书，里面放自己的public key，需要一个证书请求
  - 要使得这个证书被认可，则需要一个CA对这个证书进行签名，我们用上面的CA的private key对他进行签名
  - 其实这个过程是popsuper1982这个普通网站，将自己的请求发送给CA后，由CA来做的，只有CA才有自己的private key
    - certtool --generate-certificate --template popsuper1982_server_template.info --load-privkey popsuper1982_server_key.pem --load-ca-certificate certificate_authority_certificate.pem --load-ca-privkey certificate_authority_key.pem --outfile popsuper1982_server_certificate.pem
  - 将证书和key拷贝到相应位置
    - cp popsuper1982_server_certificate.pem /etc/pki/libvirt/servercert.pem
    - cp popsuper1982_server_key.pem /etc/pki/libvirt/private/serverkey.pem

3.3 Libvirt: Remote Access

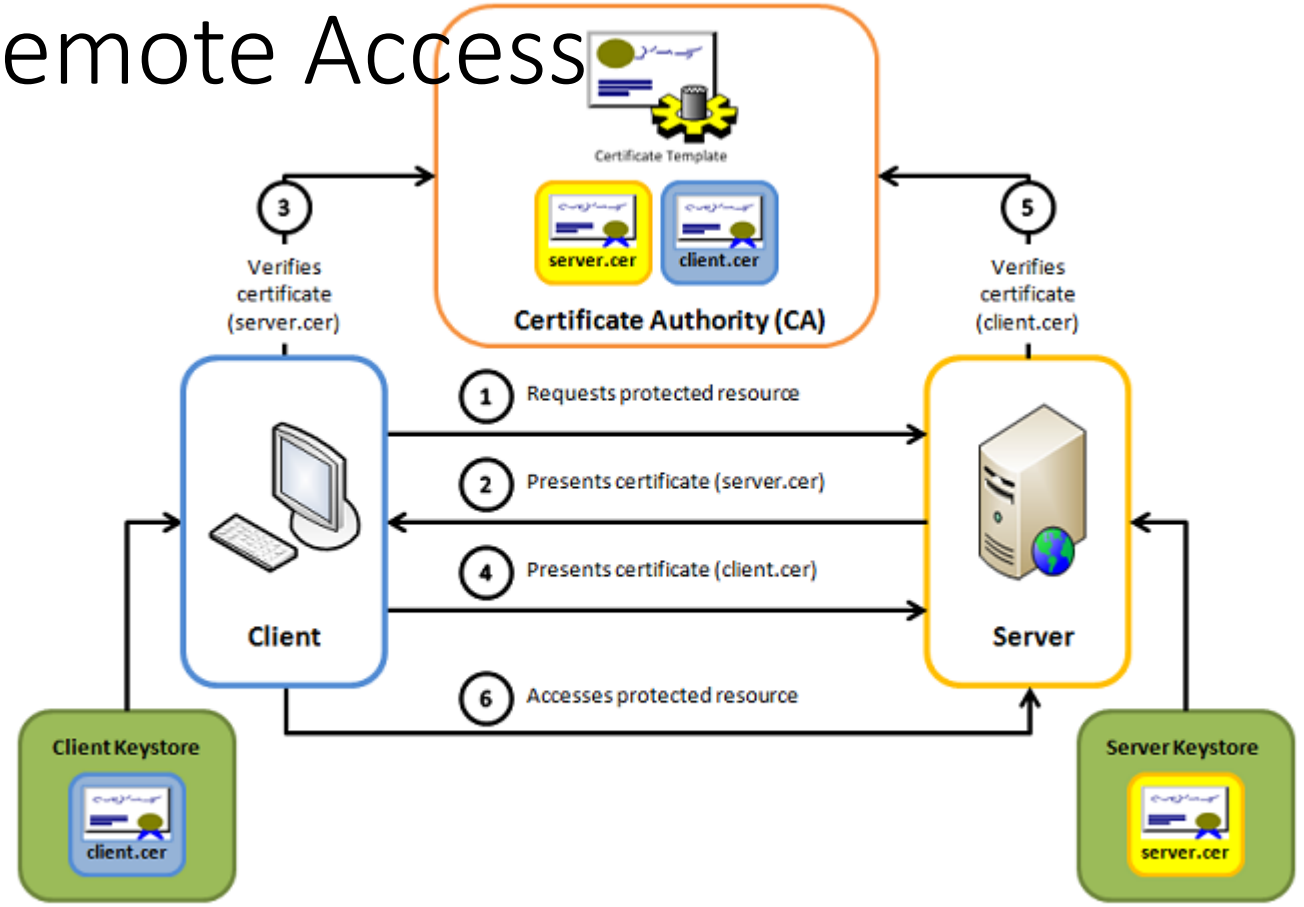
- 创建ubuntutest这台机器的key和certificate
  - ubuntutest是一个普通的机构
  - 生成一个template

```
# cat ubuntutest_client_template.info
country = CN
state = Beijing
locality = Beijing
organization = libvirt.org
cn = ubuntutest
tls_www_client
encryption_key
signing_key
```

  - 这个普通的机构需要有自己的private key
    - umask 277 && certtool --generate-privkey > ubuntutest_client_key.pem
  - 生成证书
    - certtool --generate-certificate --template ubuntutest_client_template.info --load-privkey ubuntutest_client_key.pem --load-ca-certificate certificate_authority_certificate.pem --load-ca-privkey certificate_authority_key.pem --outfile ubuntutest_client_certificate.pem
  - 将证书拷贝到相应的位置
    - scp certificate_authority_certificate.pem openstack@ubuntutest:/etc/pki/CA/cacert.pem
    - scp ubuntutest_client_certificate.pem openstack@ubuntutest:/etc/pki/libvirt/clientcert.pem
    - scp ubuntutest_client_key.pem openstack@ubuntutest:/etc/pki/libvirt/private/clientkey.pem

3.3 Libvirt: Remote Access

- 双向SSL




```
root@ubuntutest:~# tree --charset ASCII /etc/pki/
/etc/pki/
|-- CA
|   `-- cacert.pem
|-- libvirt
|   |-- clientcert.pem
|   `-- private
|       `-- clientkey.pem
`-- nssdb -> /var/lib/nssdb

4 directories, 3 files
```

Mutual SSL authentication / Certificate based mutual authentication

```
root@popsuper1982:~# tree --charset ASCII /etc/pki/
/etc/pki/
|-- CA
|   `-- cacert.pem
|-- libvirt
|   |-- private
|   |   `-- serverkey.pem
|   `-- servercert.pem
`-- nssdb -> /var/lib/nssdb

4 directories, 3 files
```

3.3 Libvirt: Remote Access

- 证书链验证过程
  - 假设我是一个普通用户，要访问网站popsuper1982, 通过https协议
  - popsuper1982会用自己的private key加密数据，传给我，我接收到数据后，需要用popsuper1982的public key进行解密
  - 这个时候popsuper1982会把它的certificate通过浏览器传给我，里面就有public key。
  - 但是我怎么能够相信这个certificate呢？我不相信，需要更高的权威机构
  - 我查看popsuper1982的certificate，发现这个certificate是由CA进行签发的，所谓签发，也即用CA的private key进行的签名。所以我要从CA的网站上下载它的certificate，里面包含public key，我们用这个public key来解签名
  - CA可以证明popsuper1982的certificate是一个对的certificate，里面的public key是可以信任的，你是可以用它来解密popsuper1982发来的数据。
  - 但是问题来了，我能相信CA这个机构么？
  - 我查看CA的certificate，发现它是自签名的，也即是一个root certificate，没有一个更高的机构为它证明了。
  - 如果我相信CA这个机构，那么我解密popsuper1982的数据，接着进行我们之间的信息交互。
  - 如果我不相信CA这个机构，在交互到此为止。
  - 当然有的时候，我们发现CA这个机构的证书是更高权威机构签名的，比如是BeijingPoliceOffice签发的，如果你相信他，你就可以进行访问。这就是证书链。

3.3 Libvirt: Remote Access

- TCP
  - 创建了这些key和certificate后， libvirt重新启动起来了
  - 在ubuntutest上运行virsh -c qemu+tcp://192.168.57.1/system list

```
root@ubuntutest:~# virsh -c qemu+tcp://192.168.57.1/system list
 Id    Name                           State
----------------------------------------------------
 26    ubuntutest                     running
```

3.3 Libvirt: Remote Access

- TLS
  - 在/etc/libvirt/libvirtd.conf中有下面的配置项
    - listen_tls = 1
    - tls_port = "16514"
    - auth_tls = "none"
    - key_file = "/etc/pki/libvirt/private/serverkey.pem"
    - cert_file = "/etc/pki/libvirt/servercert.pem"
    - ca_file = "/etc/pki/CA/cacert.pem"
  - 重启libvirt
  - 在ubuntutest上进行连接，由于是SSL，需要使用hostname
    - virsh -c qemu+tls://popsuper1982/system list

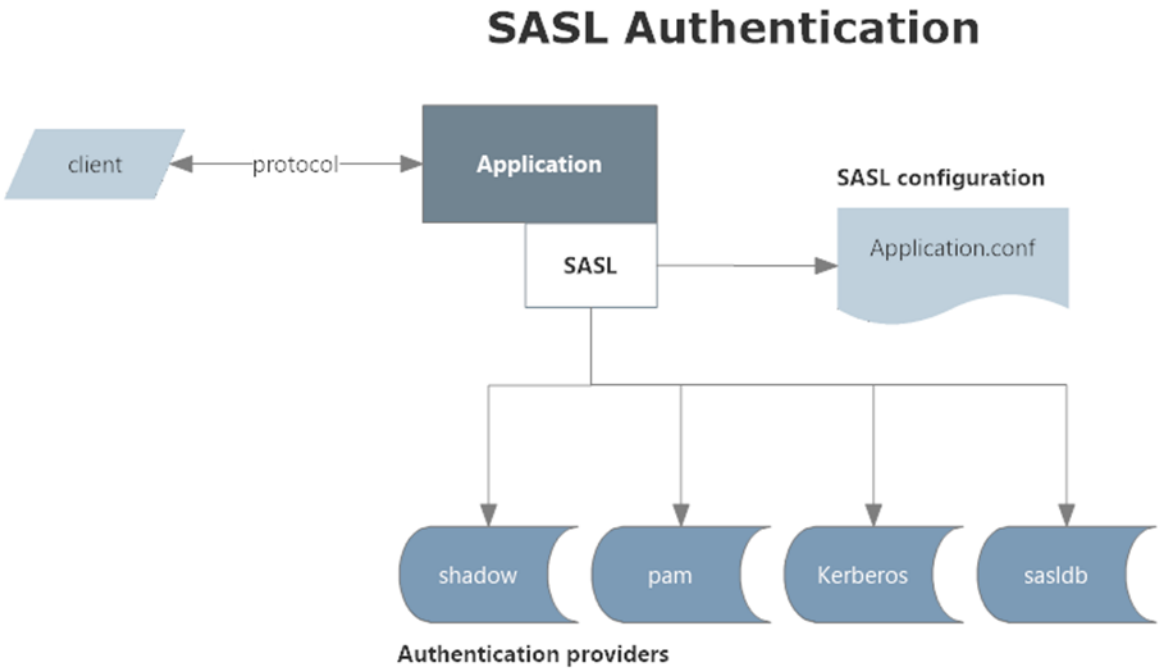
```
root@ubuntutest:~# virsh -c qemu+tls://192.168.57.1/system list
2014-10-11 10:34:56.116+0000: 2929: info : libvirt version: 1.2.2
2014-10-11 10:34:56.116+0000: 2929: warning : virNetTLSContextCheckCertificate:1140 : Certificate check failed Certificate [session] owner does not match the hostname 192.1
68.57.1
error: failed to connect to the hypervisor
error: authentication failed: Failed to verify peer's certificate
root@ubuntutest:~# virsh -c qemu+tls://popsuper1982/system list
 Id    Name                           State
----------------------------------------------------
 26    ubuntutest                     running
```

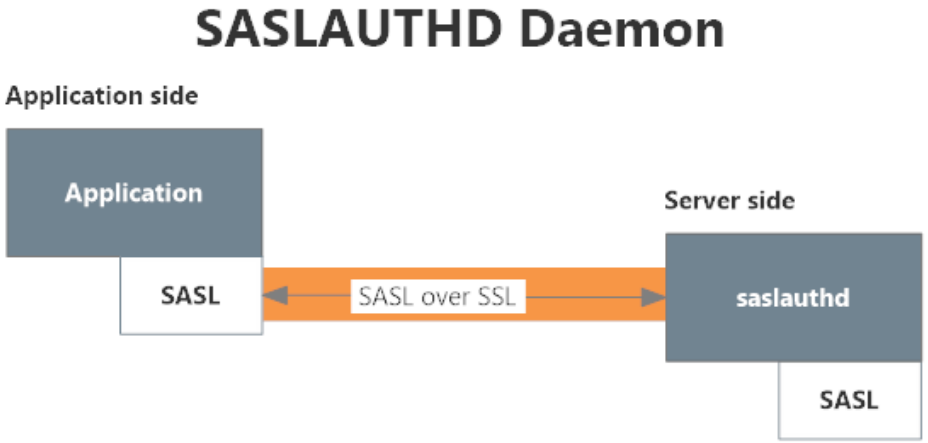
3.3 Libvirt: Remote Access

- Authentication
  - 前面使用TCP/TLS对libvirt的远程访问并不进行身份验证，下面添加身份验证过程
  - SASL(Simple Authentication and Security Layer)可使用用户名密码进行身份验证
  - Kerberos进行single-sign-on登录

3.3 Libvirt: Remote Access

- SASL是一个协议IETF Standard，将多种认证方式，多种存储方式，多种协议整合在一起
- 其中一个重要的实现是Cyrus SASL，既可以作为一个library被应用程序直接调用，也可以通过Client-Server的方式，由一个daemon来调用library进行认证



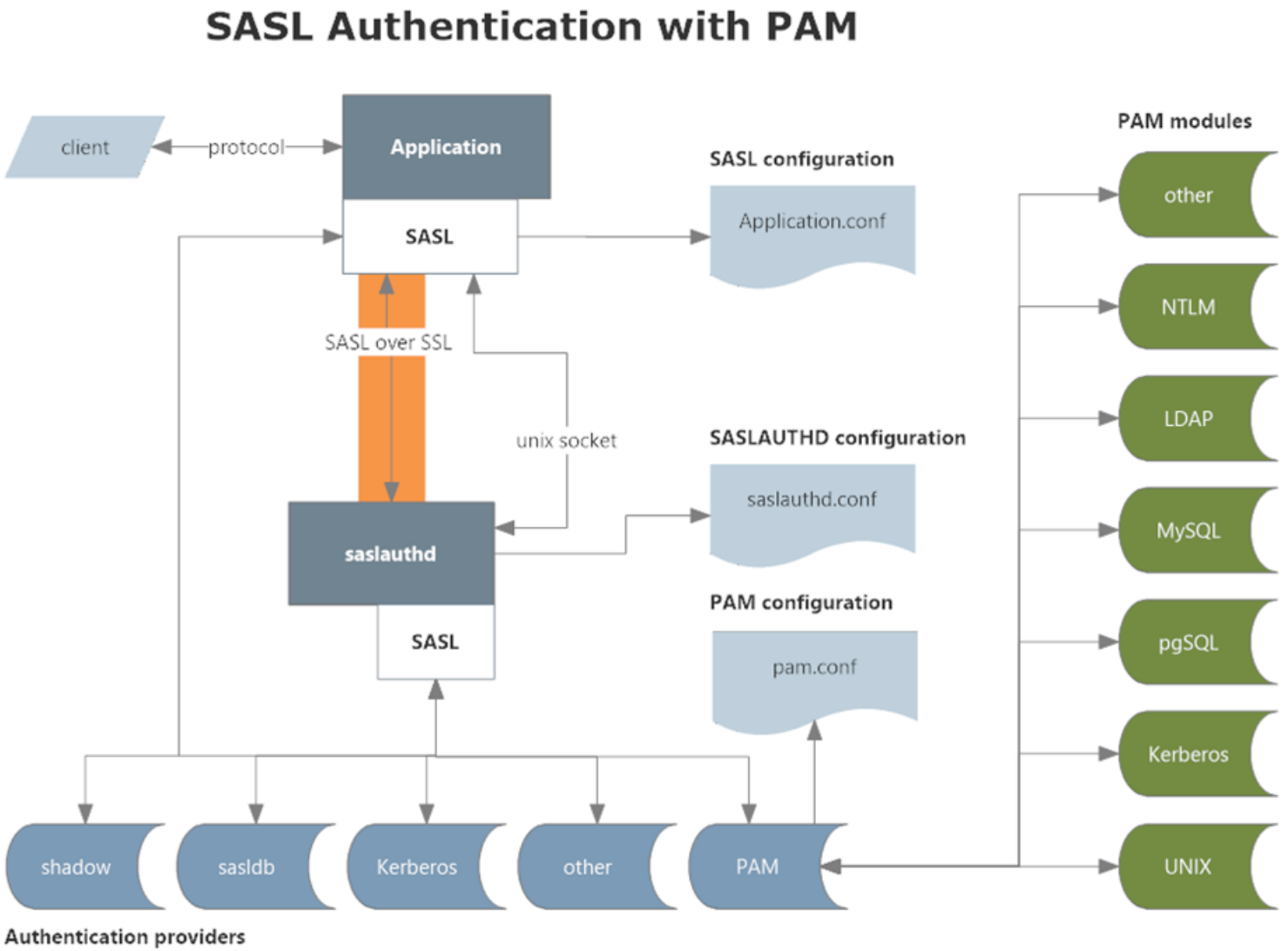


3.3 Libvirt: Remote Access

- SASL Mechanisms 认证机制，主要分三种
  - 密码验证机制(Password Verification Mechanism)：PLAIN使用明码或是BASE64的方式传递密码，这样很不安全，需要使用外部的方式进行加密，比如不是基于TCP，而是基于TLS
  - 共享密钥机制(Shared Secret Mechanism)：DIGEST-MD5，密码明文并不在网络上传输，Client和Server共享相同的密钥，通过MD5的机制进行加密
  - Kerberos Mechanism：GSSAPI来调用kerberos

3.3 Libvirt: Remote Access

- SASL 数据存储
  - passwd: 使用linux的/etc/passwd来进行用户验证
  - shadow: 使用linux的/etc/shadow文件进行用户验证
  - Kerberos
  - PAM: pluggable authentication module
  - sasldb：是一个本地的berkeleydb，来保存用户名密码



3.3 Libvirt: Remote Access

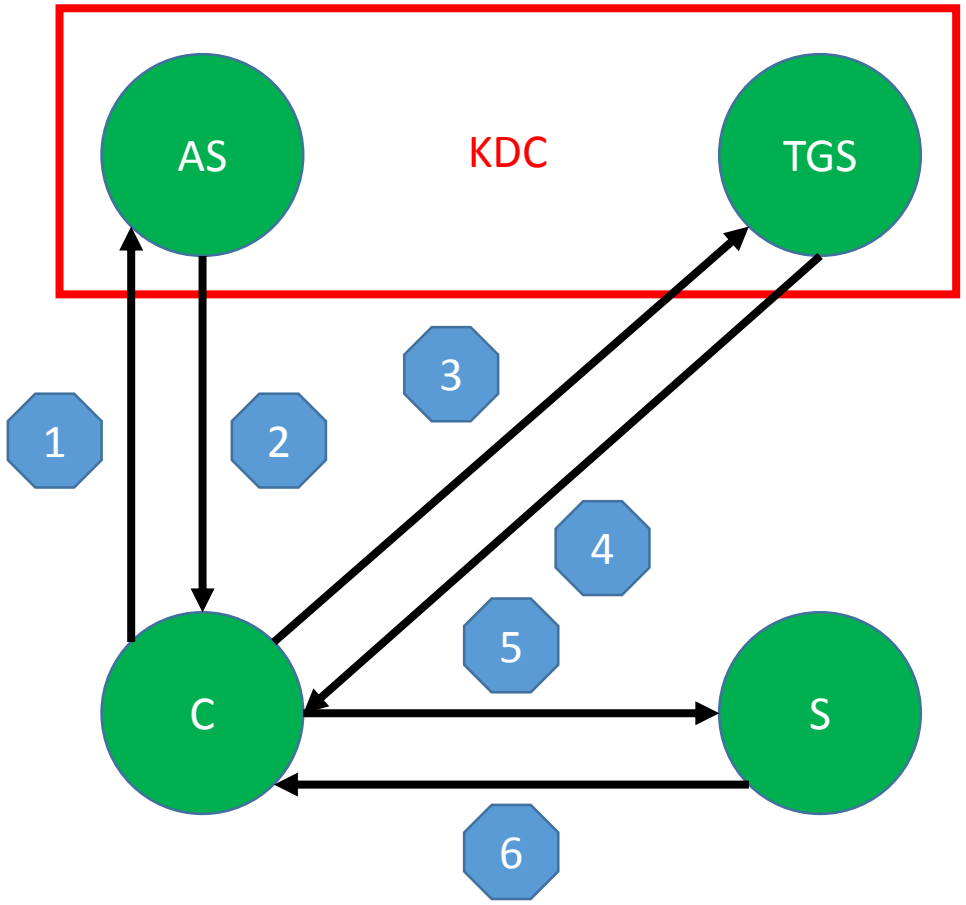
- 配置SASL
  - 修改配置文件/etc/libvirt/libvirtd.conf
    - auth_tcp = "sasl“
    - auth_tls = "sasl“
  - 重启libvirt
  - 修改配置文件/etc/sasl2/libvirt.conf
    - mech_list: digest-md5
    - sasldb_path: /etc/libvirt/passwd.db
  - apt-get install sasl2-bin
  - 添加用户
    - saslpasswd2 -a libvirt user1
    - 注意这里必须是libvirt，这是application的名字，只有输入libvirt，才会使用配置/etc/sasl2/libvirt.conf，结果才会写到/etc/libvirt/passwd.db
  - 在另外一台机器上连接的时候，需要输入用户名密码

```
root@popsuper1982:/home/openstack# saslpasswd2 -a libvirt user1
Password:
Again (for verification):
```

```
root@ubuntutest:/home/openstack# virsh -c qemu+tcp://192.168.57.1/system list
Please enter your authentication name: user1
Please enter your password:
 Id    Name                           State
----------------------------------------------------
 26    ubuntutest                     running

root@ubuntutest:/home/openstack# virsh -c qemu+tls://popsuper1982/system list
Please enter your authentication name: user1
Please enter your password:
 Id    Name                           State
----------------------------------------------------
 26    ubuntutest                     running
```

3.3 Libvirt: Remote Access

- Kerberos：三个角色，六个过程
  - C：客户端Client
  - S：服务器Server
  - KDC：密钥分配中心Key Distribution Center
  - AS：认证服务器Authentication Server
  - TGS：票据授权服务器Ticket Granting Server



3.3 Libvirt: Remote Access

- (1) Client -> AS
  - 客户端向认证服务器发送请求
  - 请求内容：
    - C: Client，我是谁
    - TGS: 访问哪个票据授权服务器，可以是本域的，也可以是其他域的
    - Timestamp: 时间戳
    - IP Address
  - 请求以明文发送
  - 这个过程没有任何密码以明文形式发送

3.3 Libvirt: Remote Access

- (2) AS -> Client
  - 认证服务器回复客户端
  - 回复内容
    - $\{K_{c,TGS}, T_{c,TGS}\} K_c$
    - 内容由K_c加密，它是客户的加密密钥，多由客户的密码通过哈希算法得到，认证服务器的数据库里面有客户的密钥，客户自己也有这个密钥，解开这个回复
    - 回复包含两部分，第一部分$K_{c,TGS}$是客户端和票据授权服务器之间的共享的会话(Session)密钥
    - 第二部分是访问TGS的票据Ticket $T_{c,TGS}$
    - $T_{c,TGS}$内容为$\{TGS, Client, IP\ Address, Timestamp, lifetime, K_{c,TGS}\} K_{TGS}$
  - 说明
    - 在Kerberos中，对客户的认证时统一由KDC来的，KDC是大家相信也唯一相信的系统
    - 作为Single sign-on (SSO)系统，这是唯一一次客户使用自己的密码。一旦登录，此后访问不同的服务(邮件，打印机，GIT)等，都不需要再输入密码再登录了。
    - 到这一步，客户已经登录成功了，AS的工作完成。但是客户访问哪个服务还没定，都是和TGS交互决定的。所以返回的内容中，都是和TGS交互所必须的内容
    - $K_{c,TGS}$是客户端和TGS交互的时候，用于加密的会话密钥
    - $T_{c,TGS}$是AS颁发给客户的票据，这个票据是由K_{TGS}加密的，客户不知道里面是什么，客户拿着这个票据，就可以访问TGS了，TGS知道里面是什么
    - 这相当于公安局给客户开了一个证明(比如身份证)，客户拿着这个身份证可以去银行了，客户不知道身份证词条里面是什么信息，银行一刷就知道了

3.3 Libvirt: Remote Access

- (3) Client -> TGS
  - 客户已经登录了，还没说要访问哪个服务，这一步就是请求KDC授权对服务的访问
  - 请求内容
    - S: Server，我要访问哪个服务
    - $T_{c,TGS}$: AS给客户的票据
    - 认证符Auhenticator: {Client, IP Address, timestamp} $K_{c,TGS}$: 客户将自己的信息加密
  - 说明：
    - TGS拿到AS给客户开具的票据$T_{c,TGS}$= {TGS, Client, IP Address, Timestamp, lifetime, $K_{c,TGS}$} K_{TGS}，用自己的密钥解开，得到客户的信息，和与客户交互使用的密钥$K_{c,TGS}$
    - TGS拿到认证符，用密钥$K_{c,TGS}$解密，也得到客户的信息
    - TGS通过比较，发现用户自己提供的信息，和AS提供的信息一致，则可以访问服务S
    - 这个过程相当于，你拿着公安局开具的身份证到银行，银行一刷身份证，得到了你的信息，然后跟你核对生日，固定电话等，和你提供的一致，你就可以办业务了

3.3 Libvirt: Remote Access

- (4) TGS -> Client
  - 当TGS验证结束后，将授权客户访问服务
  - 回复内容：
    - $\{K_{c,s}, T_{c,s}\} K_{c,TGS}$
    - 内容被$K_{c,TGS}$加密，可以被客户端解开
    - 里面包含两个内容，一个是$K_{c,s}$，客户端和服务器交互也是需要加密的，便使用这个会话Session密钥
    - 另一个是$T_{c,s}$ ={S, C, IP Address, timestamp, lifetime, $K_{c,s}$} K_s，在这里TGS又成了公安局，为了能够让客户访问服务器，向客户开具票据，将来客户访问服务器的时候，需要拿着这个票据

3.3 Libvirt: Remote Access

- (5) Client -> Server
  - 客户访问服务器
  - 请求内容：
    - S: Server
    - $T_{c,s}$: TGS给客户的票据
    - 认证符Auhenticator: {Client, IP Address, timestamp} $K_{c,s}$: 客户自己的信息，由会话密钥加密
  - 说明：
    - 当服务器拿到TGS给客户的票据$T_{c,s}$ ={S, C, IP Address, timestamp, lifetime, $K_{c,s}$} K_s ，用自己的密钥解开，得到客户的信息，以及将来和客户交互使用的会话密钥$K_{c,s}$
    - 服务器拿到客户的认证符，用$K_{c,s}$解密后，得到客户的信息
    - 信息比较一致，则客户可以访问服务器

3.3 Libvirt: Remote Access

- (6) Server -> Client
  - 服务器回复客户端
  - 回复内容：
    - {timestamp+1} $K_{c,s}$
  - 说明：
    - 客户端得到回复后，只有真正的客户，才能用$K_{c,s}$才能解开
    - 通过验证时间戳，客户端也确认服务器就是要想访问的服务器，实现了双向认证

3.3 Libvirt: Remote Access

- Access Control: Polkit

3.4 Libvirt: Control Group

- Control Group又称cgroup，用于控制进程的资源使用：CPU, Disk I/O, Memory, Network等
- 几个重要的概念：
  - 任务：对应于进程，将进程ID放到cgroup里面，就算cgroup的一个任务，受cgroup控制
  - 层次：
    - cgroup是分层次的，子cgroup会继承父cgroup的设置
    - Cgroup的管理通常使用file system的接口，一个cgroup会被mount成一个cgroup的文件系统，则层次结构对应于文件夹结构
  - 子系统：系统的资源被不同的子系统控制
    - blkio -- 为块设备设定输入/输出限制，比如物理设备(磁盘，固态硬盘，USB等等)。
    - cpu -- 使用调度程序提供对CPU的cgroup任务访问。
    - cpuacct -- 自动生成cgroup中任务所使用的CPU报告。
    - cpuset -- 为cgroup中的任务分配独立CPU (在多核系统)和内存节点。
    - devices -- 可允许或者拒绝cgroup中的任务访问设备。
    - freezer -- 挂起或者恢复cgroup中的任务。
    - memory -- 设定cgroup中任务使用的内存限制，并自动生成由那些任务使用的内存资源报告。
    - net_cls -- 使用等级识别符(classid)标记网络数据包，可允许Linux流量控制程序tc识别从 具体cgroup中生成的数据包。
    - ns -- 名称空间子系统。

3.4 Libvirt: Control Group

3.4 Libvirt: Control Group

两个libvirt进程

```
root@popsuper1982:/home/openstack/images# virsh list
 Id    Name                           State
----------------------------------------------------
 29    ubuntutest                     running
 30    ubuntutest2                    running
```

```
root@popsuper1982:/home/openstack/images# cat /sys/fs/cgroup/cpu/machine/ubuntutest.libvirt-qemu/vcpu0/tasks
21282
root@popsuper1982:/home/openstack/images# cat /sys/fs/cgroup/cpu/machine/ubuntutest2.libvirt-qemu/vcpu0/tasks
21320
```

树形结构

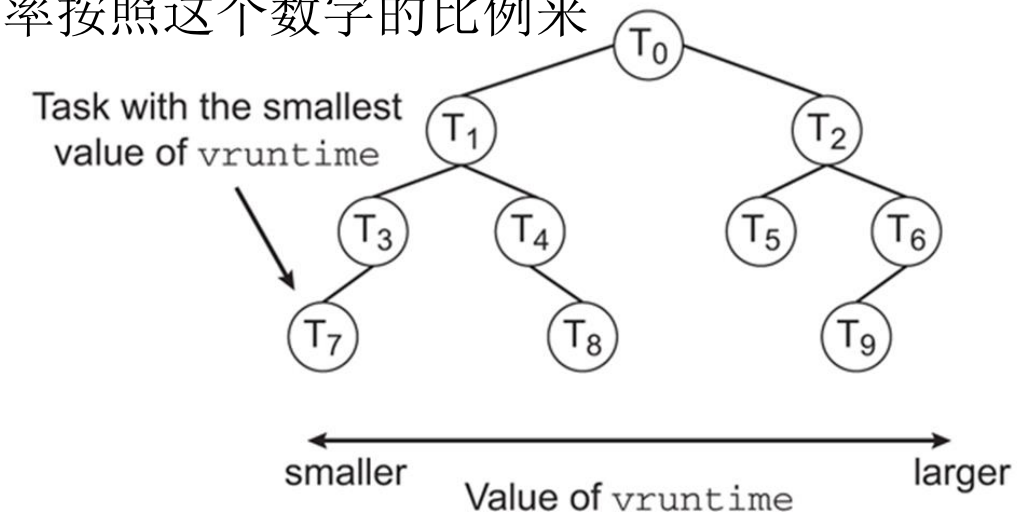
```
root@popsuper1982:/home/openstack/images# tree --charset ASCII /sys/fs/cgroup/cpu
/sys/fs/cgroup/cpu
|-- cgroup.clone_children
|-- cgroup.event_control
|-- cgroup.procs
|-- cgroup.sane_behavior
|-- cpu.cfs_period_us
|-- cpu.cfs_quota_us
|-- cpu.shares
|-- cpu.stat
|-- machine
|   |-- cgroup.clone_children
|   |-- cgroup.event_control
|   |-- cgroup.procs
|   |-- cpu.cfs_period_us
|   |-- cpu.cfs_quota_us
|   |-- cpu.shares
|   |-- cpu.stat
|   |-- notify_on_release
|   |-- tasks
|   |-- ubuntutest2.libvirt-qemu
|   |   |-- cgroup.clone_children
|   |   |-- cgroup.event_control
|   |   |-- cgroup.procs
|   |   |-- cpu.cfs_period_us
|   |   |-- cpu.cfs_quota_us
|   |   |-- cpu.shares
|   |   |-- cpu.stat
|   |   |-- emulator
|   |   |   |-- cgroup.clone_children
|   |   |   |-- cgroup.event_control
|   |   |   |-- cgroup.procs
|   |   |   |-- cpu.cfs_period_us
|   |   |   |-- cpu.cfs_quota_us
|   |   |   |-- cpu.shares
|   |   |   |-- cpu.stat
|   |   |   |-- notify_on_release
|   |   |   `-- tasks
|   |   |-- notify_on_release
|   |   |-- tasks
|   |   `-- vcpu0
|   |       |-- cgroup.clone_children
|   |       |-- cgroup.event_control
|   |       |-- cgroup.procs
|   |       |-- cpu.cfs_period_us
|   |       |-- cpu.cfs_quota_us
|   |       |-- cpu.shares
|   |       |-- cpu.stat
|   |       |-- notify_on_release
|   |       `-- tasks
|   `-- ubuntutest.libvirt-qemu
|       |-- cgroup.clone_children
|       |-- cgroup.event_control
|       |-- cgroup.procs
|       |-- cpu.cfs_period_us
|       |-- cpu.cfs_quota_us
|       |-- cpu.shares
|       |-- cpu.stat
|       |-- emulator
|       |   |-- cgroup.clone_children
|       |   |-- cgroup.event_control
|       |   |-- cgroup.procs
|       |   |-- cpu.cfs_period_us
|       |   |-- cpu.cfs_quota_us
|       |   |-- cpu.shares
|       |   |-- cpu.stat
|       |   |-- notify_on_release
|       |   `-- tasks
|       |-- notify_on_release
|       |-- tasks
|       `-- vcpu0
|           |-- cgroup.clone_children
|           |-- cgroup.event_control
|           |-- cgroup.procs
|           |-- cpu.cfs_period_us
|           |-- cpu.cfs_quota_us
|           |-- cpu.shares
|           |-- cpu.stat
|           |-- notify_on_release
|           `-- tasks
```

3.4 Libvirt: Control Group

- 安装：
  - apt-get install libcgroup cgroup-bin
- 创建cgroup文件系统并且attach子系统
  - mount -t cgroup -o cpu,cpuset,memory cpu_and_mem /cgroup/cpu_and_mem
- 对cgroup的设置可以通过文件系统
  - 创建：mkdir /cgroup/cpu/testcg
  - 删除：rmdir /cgroup/cpu/testcg
  - 修改参数：echo 2048 > /cgroup/cpu/testcg/cpu.shares
  - 将任务加入cgroup: echo 1234 > /cgroup/cpu/testcg/tasks
  - 在cgroup中执行命令：
    - cgexec -g cpu:testcg lynx http://www.redhat.com
    - 将命令行作为task: echo $$ > /cgroup/testcg/tasks，然后在命令行里面执行lynx http://www.redhat.com
- 查看所有的cgroup
  - lscgroup
- 查看所有的子系统：
  - lssubsys -am
- 查看一个进程属于哪个cgroup:
  - ps -O cgroup

3.4 Libvirt: Control Group

- 对CPU的控制
  - CFS (Completely Fair Scheduler)：
    - 将所有可运行的任务放入一棵红黑树，key是vruntime
    - 红黑树是平衡的，左面的任务得到的时间少，右面的任务得到的时间多，最左面的任务有最高的优先级
    - cfs_period_us的意思是cgroup对CPU的调度的干预周期，cfs_quota_us是指则一个周期内这个进程得到的时间片长度。比如cfs_period_us=100000说明100毫秒cgroup进行一次干预，cfs_quota_us=25000表示在100毫秒里面，这个进程能够得到25毫秒的时间片，如果对cgroup进行的修改，则要等到下个100毫秒才起作用
    - cpu.shares是一个相对值，进程之间的cpu使用率按照这个数字的比例来



3.4 Libvirt: Control Group

- 对CPU的控制
  - 查看cpuset
    - cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,relatime,cpuset)
    - cat ubuntutest.libvirt-qemu/cpuset.cpus #0-3
    - cat ubuntutest2.libvirt-qemu/cpuset.cpus #0-3
    - 两个VM都同时使用四个CPU，为了查看CPU调度，我们使用一个CPU
    - 关闭VM，设置cpuset.cpus

```
root@popsuper1982:/sys/fs/cgroup/cpuset/machine# echo 0 > cpuset.cpus
root@popsuper1982:/sys/fs/cgroup/cpuset/machine# cat cpuset.cpus
0
```

    - 重启VM

```
root@popsuper1982:/sys/fs/cgroup/cpuset/machine# cat ubuntutest.libvirt-qemu/cpuset.cpus
0
root@popsuper1982:/sys/fs/cgroup/cpuset/machine# cat ubuntutest2.libvirt-qemu/cpuset.cpus
0
```

3.4 Libvirt: Control Group

```
root@ubuntutest2:/home/openstack# cat loop.py
#!/usr/bin/python
import time
i=0
while True:
#       time.sleep(10)
        i=i+1
```

- 对CPU的控制
  - 在两台虚拟机上创建一个无限循环的python程序
  - 没有设置cgroup的时候，两台虚拟机同时运行这个程序

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
22743 libvirt+  20   0 3511136 360628   9564 S  50.2  9.4   0:21.45 qemu-system-x86
22808 libvirt+  20   0 3511088 306368   9564 S  49.9  8.0   0:33.81 qemu-system-x86
```

  - 设置cpu.shares
    - 查看现在的设置

```
root@popsuper1982:/sys/fs/cgroup/cpu/machine# cat ubuntutest.libvirt-qemu/cpu.shares
1024
root@popsuper1982:/sys/fs/cgroup/cpu/machine# cat ubuntutest2.libvirt-qemu/cpu.shares
1024
```

    - 修改ubuntutest2的设置
      - echo 4096 > ubuntutest2.libvirt-qemu/cpu.shares

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
22808 libvirt+  20   0 3511088 308392   9564 S  79.8  8.0   3:27.32 qemu-system-x86
22743 libvirt+  20   0 3511136 360628   9564 S  20.3  9.4   3:07.42 qemu-system-x86
```

3.4 Libvirt: Control Group

- 对CPU的控制
  - 也可以通过virsh设置和查看
  - virsh schedinfo ubuntutest --set cpu_shares=2048

```
root@popsuper1982:/sys/fs/cgroup/cpu/machine# virsh schedinfo ubuntutest
Scheduler      : posix
cpu_shares     : 2048
vcpu_period    : 100000
vcpu_quota     : -1
emulator_period: 100000
emulator_quota : -1

root@popsuper1982:/sys/fs/cgroup/cpu/machine# virsh schedinfo ubuntutest2
Scheduler      : posix
cpu_shares     : 4096
vcpu_period    : 100000
vcpu_quota     : -1
emulator_period: 100000
emulator_quota : -1
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
22808 libvirt+  20   0 3511088 338648   9564 S  66.5  8.8  12:00.69 qemu-system-x86
22743 libvirt+  20   0 3511136 360624   9564 S  33.6  9.4   6:32.03 qemu-system-x86
```

3.4 Libvirt: Control Group

- 对CPU的控制
  - 设置vcpu_quota

```
root@popsuper1982:/sys/fs/cgroup/cpu/machine# virsh schedinfo ubuntutest --set vcpu_quota=10000
Scheduler      : posix
cpu_shares     : 2048
vcpu_period    : 100000
vcpu_quota     : 10000
emulator_period: 100000
emulator_quota : -1

root@popsuper1982:/sys/fs/cgroup/cpu/machine# virsh schedinfo ubuntutest2 --set vcpu_quota=50000
Scheduler      : posix
cpu_shares     : 4096
vcpu_period    : 100000
vcpu_quota     : 50000
emulator_period: 100000
emulator_quota : -1
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
22808 libvirt+  20   0 3511088 338648   9564 S  50.2  8.8  20:42.75 qemu-system-x86
22743 libvirt+  20   0 3511136 360624   9564 S  10.0  9.4  10:45.34 qemu-system-x86
```

3.4 Libvirt: Control Group

- 对block I/O的控制
  - 查看major number和minor number

```
root@popsuper1982:/dev# ls -l /dev/sda*
brw-rw---- 1 root disk 8, 0 Sep 10 10:46 /dev/sda
brw-rw---- 1 root disk 8, 1 Sep 10 10:46 /dev/sda1
brw-rw---- 1 root disk 8, 2 Sep 10 10:46 /dev/sda2
brw-rw---- 1 root disk 8, 5 Sep 10 10:46 /dev/sda5
```

  - 相对值：范围在**100**到**1000**
    - echo 8:0 500 > blkio.weight_device
  - 绝对值(bytes per second)：
    - echo "8:0 10485760" > /cgroup/blkio/testcg/blkio.throttle.read_bps_device
    - echo "8:0 10485760" > /cgroup/blkio/testcg/blkio.throttle.write_bps_device
  - 使用virsh

```
root@popsuper1982:/dev# virsh blkiotune ubuntutest
weight         : 500
device_weight  :
device_read_iops_sec:
device_write_iops_sec:
device_read_bytes_sec:
device_write_bytes_sec:
```

  - 使用iotop -P -o查看速度

3.4 Libvirt: Control Group

- 对block I/O的控制
  - 首先cache='none'

```
<disk type='file' device='disk'>
    <driver name='qemu' type='qcow2' cache='none'/>
    <source file='/home/openstack/images/ubuntutest2.qcow2'/>
    <target dev='vda' bus='virtio'/>
</disk>
```

  - 设置写入速度
    - echo "8:0 10485760" > ubuntutest.libvirt-qemu/blkio.throttle.write_bps_device
    - echo "8:0 20971520" > ubuntutest.libvirt-qemu/blkio.throttle.write_bps_device

```
root@ubuntutest:/home/openstack# dd if=/dev/zero of=world.img bs=1024k count=1024 oflag=direct
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 102.459 s, 10.5 MB/s
root@ubuntutest:/home/openstack# dd if=/dev/zero of=world.img bs=1024k count=1024 oflag=direct
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 51.2234 s, 21.0 MB/s
```

3.4 Libvirt: Control Group

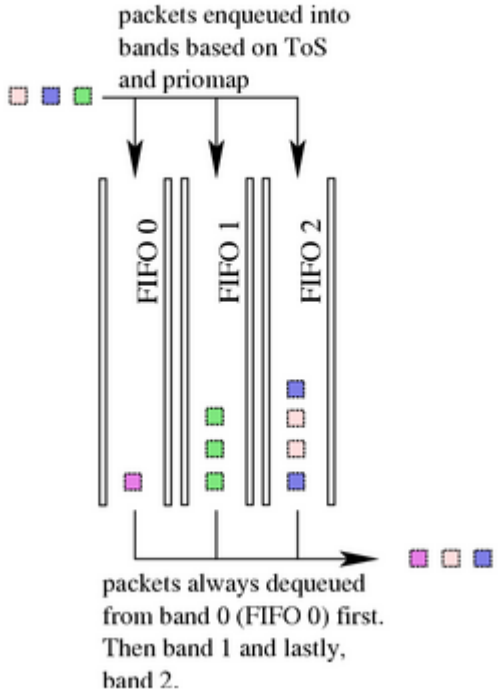
- Network
  - 对于virsh命令，当前net_cls还不支持
  - 可以通过TC直接控制网卡的流量

3.4 Libvirt: Control Group

- Network
  - **Classless Queuing Disciplines**
  - 默认为pfifo_fast

```
root@popsuper1982:/home/openstack# ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 68:b5:99:ef:5c:ac brd ff:ff:ff:ff:ff:ff
```

pfifo_fast queuing discipline




```
   0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|                 |                       |     |
|   PRECEDENCE    |          TOS          | MBZ |
|                 |                       |     |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

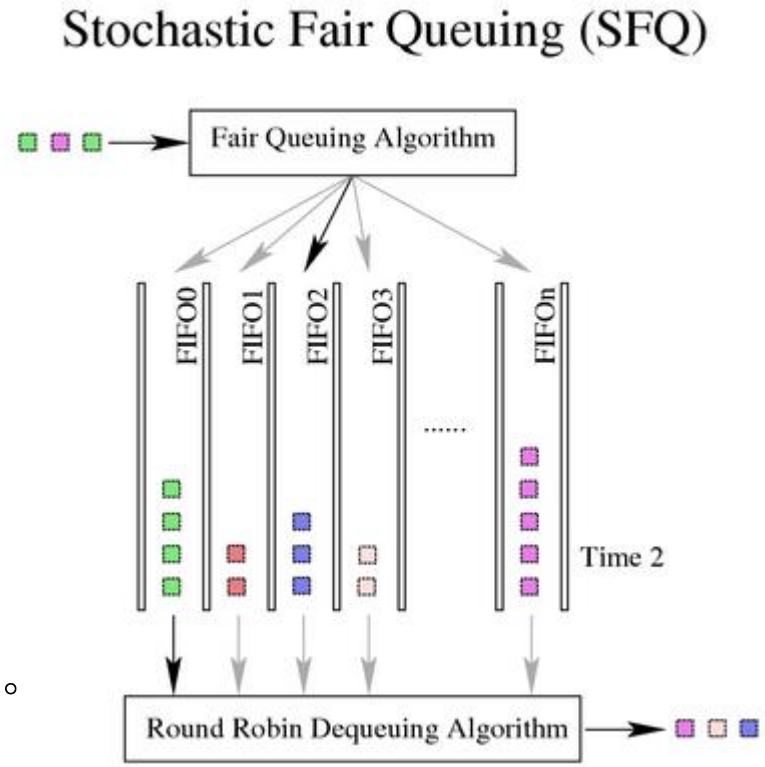
Binary	Decimcal	Meaning
1000	8	Minimize delay (md)
0100	4	Maximize throughput (mt)
0010	2	Maximize reliability (mr)
0001	1	Minimize monetary cost (mmc)
0000	0	Normal Service

TOS	Bits	Means	Linux Priority	Band
0x0	0	Normal Service	0 Best Effort	1
0x2	1	Minimize Monetary Cost	1 Filler	2
0x4	2	Maximize Reliability	0 Best Effort	1
0x6	3	mmc+mr	0 Best Effort	1
0x8	4	Maximize Throughput	2 Bulk	2
0xa	5	mmc+mt	2 Bulk	2
0xc	6	mr+mt	2 Bulk	2
0xe	7	mmc+mr+mt	2 Bulk	2
0x10	8	Minimize Delay	6 Interactive	0
0x12	9	mmc+md	6 Interactive	0
0x14	10	mr+md	6 Interactive	0
0x16	11	mmc+mr+md	6 Interactive	0
0x18	12	mt+md	4 Int. Bulk	1
0x1a	13	mmc+mt+md	4 Int. Bulk	1
0x1c	14	mr+mt+md	4 Int. Bulk	1
0x1e	15	mmc+mr+mt+md	4 Int. Bulk	1

```
root@popsuper1982:/home/openstack# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
```

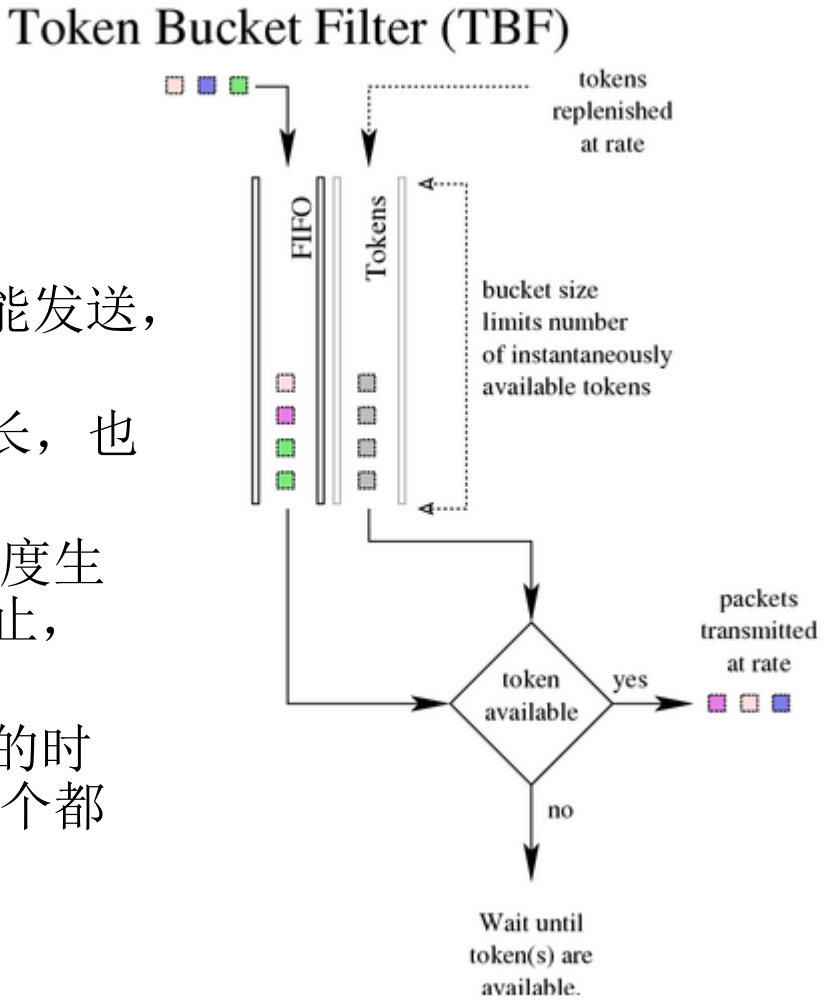
3.4 Libvirt: Control Group

- Network
  - SFQ, Stochastic Fair Queuing
    - 有很多的FIFO的队列，TCP Session或者UDP stream会被分配到某个队列。包会RoundRobin的从各个队列中取出发送。
    - 这样不会一个Session占据所有的流量。
    - 但不是每一个Session都有一个队列，而是有一个Hash算法，将大量的Session分配到有限的队列中。
    - 这样两个Session会共享一个队列，也有可能互相影响。
    - Hash函数会经常改变，从而session不会总是相互影响。



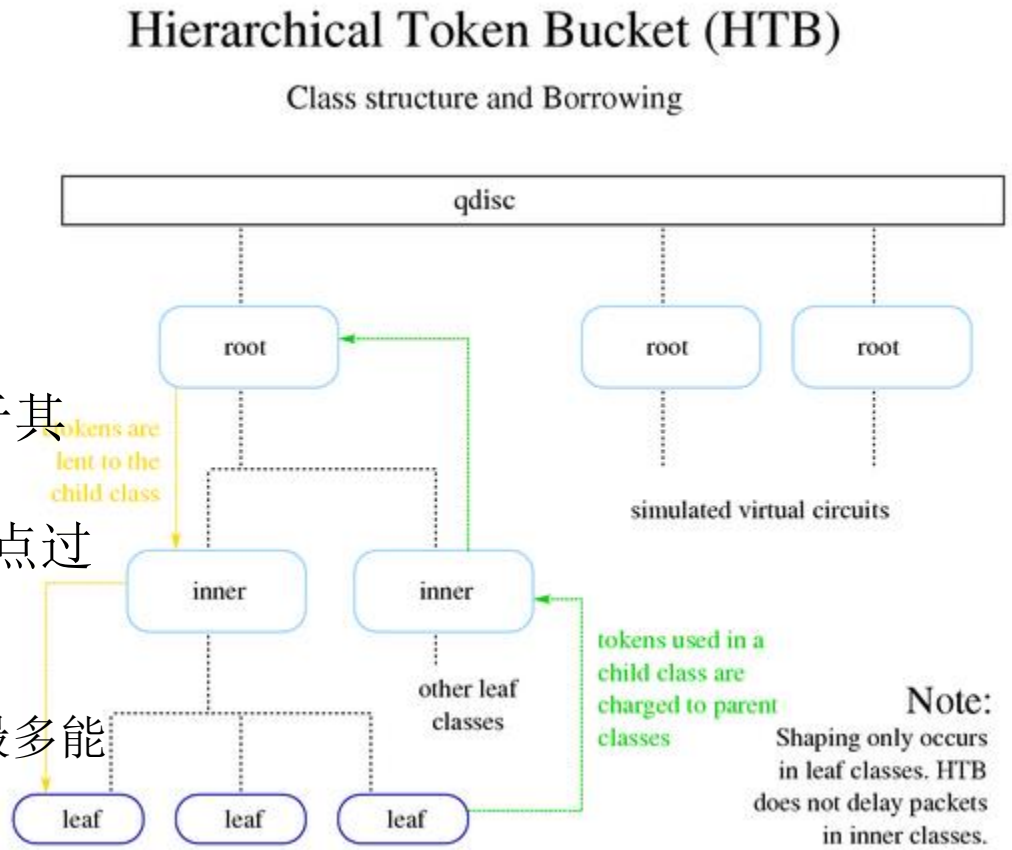
3.4 Libvirt: Control Group

- Network
  - TBF, Token Bucket Filter
    - 两个概念Tokens and buckets
    - 所有的包排成队列进行发送，但不是到了队头就能发送，而是需要拿到Token才能发送
    - Token根据设定的速度rate生成，所以即便队列很长，也是按照rate进行发送的
    - 当没有包在队列中的时候，Token还是以既定的速度生成，但是不是无限累积的，而是放满了buckets为止，篮子的大小常用burst/buffer/maxburst来设定
    - Buckets会避免下面的情况：当长时间没有包发送的时候，积累了大量的Token，突然来了大量的包，每个都能得到Token，造成瞬间流量大增



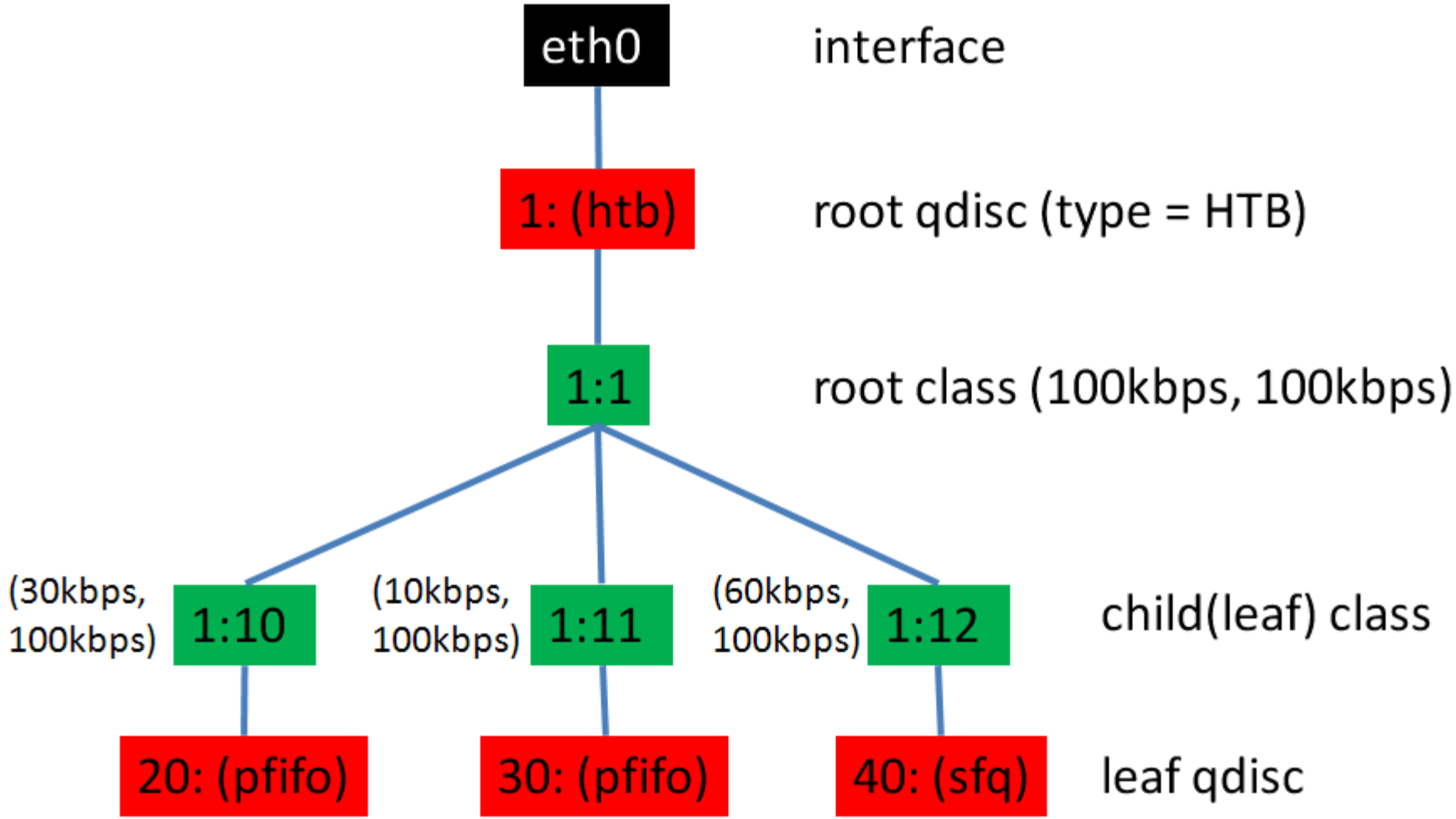
3.4 Libvirt: Control Group

- **Network**
  - **Classful Queuing Disciplines**
  - HTB, Hierarchical Token Bucket
    - Shaping：仅仅发生在叶子节点，依赖于其他的Queue
    - Borrowing: 当网络资源空闲的时候，借点过来为我所用
      - Rate：设定的发送速度
      - Ceil：最大的速度，和rate之间的差是最多能向别人借多少



type of class	class state	HTB internal state	action taken
leaf	< rate	HTB_CAN_SEND	Leaf class will dequeue queued bytes up to available tokens (no more than burst packets)
leaf	> rate, < ceil	HTB_MAY_BORROW	Leaf class will attempt to borrow tokens/ctokens from parent class. If tokens are available, they will be lent in quantum increments and the leaf class will dequeue up to cburst bytes
leaf	> ceil	HTB_CANT_SEND	No packets will be dequeued. This will cause packet delay and will increase latency to meet the desired rate.
inner, root	< rate	HTB_CAN_SEND	Inner class will lend tokens to children.
inner, root	> rate, < ceil	HTB_MAY_BORROW	Inner class will attempt to borrow tokens/ctokens from parent class, lending them to competing children in quantum increments per request.
inner, root	> ceil	HTB_CANT_SEND	Inner class will not attempt to borrow from its parent and will not lend tokens/ctokens to children classes.

3.4 Libvirt: Control Group

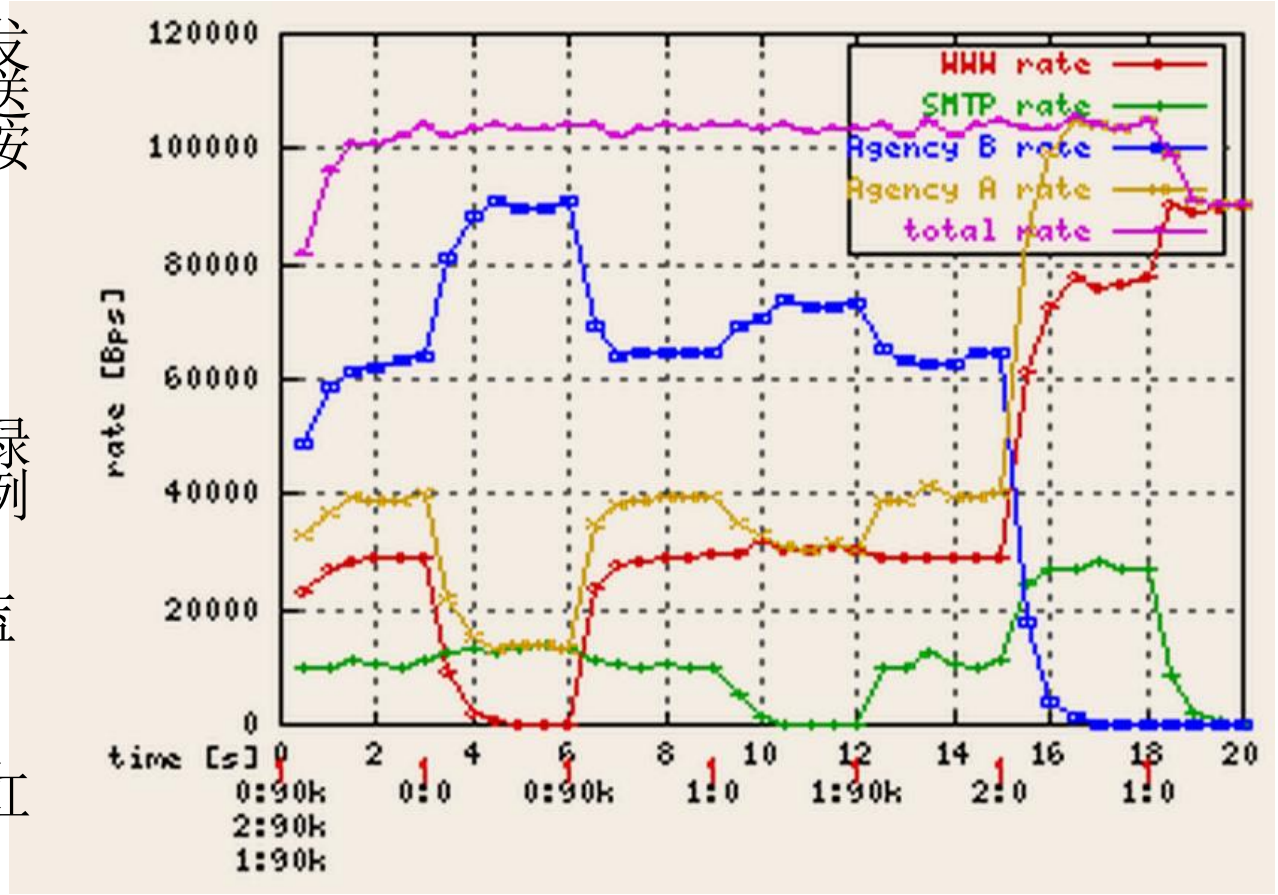


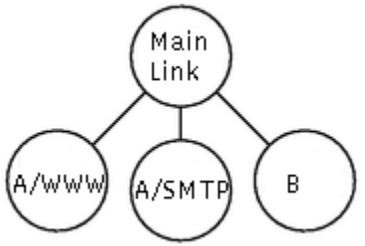
3.4 Libvirt: Control Group

创建一个HTB的qdisc在eth0上，句柄为1:，default 12表示默认发送给1:12
tc qdisc add dev eth0 root handle 1: htb default 12
创建一个root class，然后创建几个子class
同一个root class下的子类可以相互借流量，如果直接不在qdisc下面创建一个root class，而是直接创建三个class，他们之间是不能相互借流量的。
tc class add dev eth0 parent 1: classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 30kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:11 htb rate 10kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 60kbps ceil 100kbps
创建叶子qdisc，分别为fifo和sfq
tc qdisc add dev eth0 parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev eth0 parent 1:11 handle 30: pfifo limit 5
tc qdisc add dev eth0 parent 1:12 handle 40: sfq perturb 10
设定规则：从1.2.3.4来的，发送给port 80的包，从1:10走；其他从1.2.3.4发送来的包从1:11走；其他的走默认
tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip src 1.2.3.4 match ip dport 80 0xffff flowid 1:10
tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip src 1.2.3.4 flowid 1:11

3.4 Libvirt: Control Group

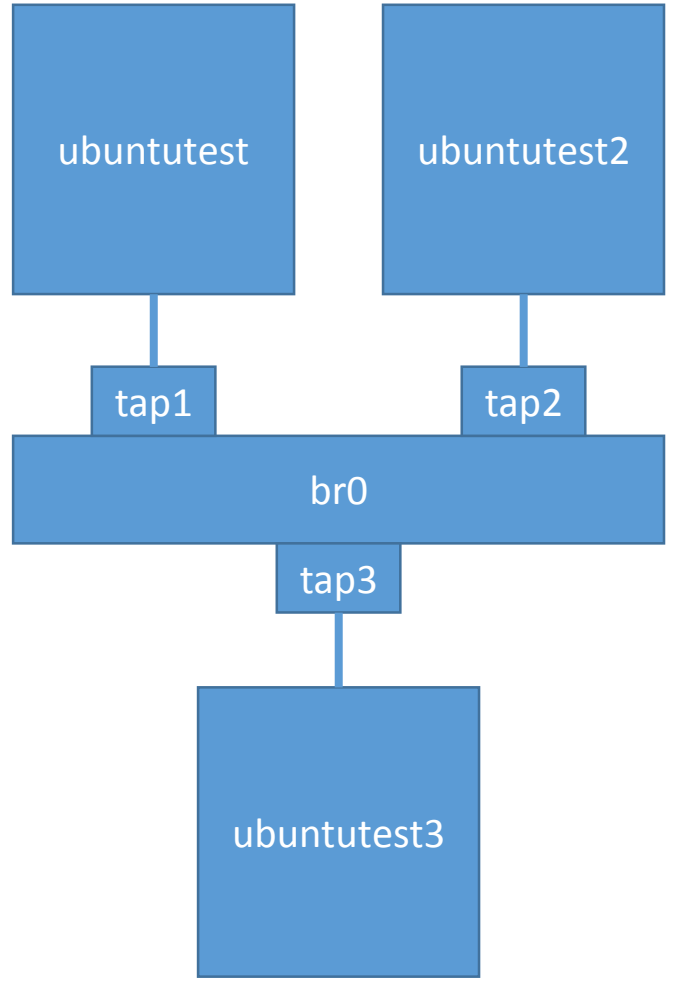
- 时间0的时候，0,1,2都以90k的速度发送数据，在时间3的时候，将0的发送停止，红色的线归零，剩余的流量按照比例分给了蓝色的和绿色的线。
- 在时间6的时候，将0的发送重启为90k，则蓝色和绿色的流量返还给红色的流量。
- 在时间9的时候，将1的发送停止，绿色的流量为零，剩余的流量按照比例分给了蓝色和红色。
- 在时间12,将1的发送恢复，红色和蓝色返还流量。
- 在时间15，将2的发送停止，蓝色流量为零，剩余的流量按照比例分给红色和绿色。
- 在时间19，将1的发送停止，绿色的流量为零，所有的流量都归了红色。





3.4 Libvirt: Control Group

```
tc qdisc add dev tap3 root handle 1: htb
tc class add dev tap3 parent 1: classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev tap3 parent 1:1 classid 1:10 htb rate 20kbps ceil 100kbps
tc class add dev tap3 parent 1:1 classid 1:11 htb rate 80kbps ceil 100kbps
tc qdisc add dev tap3 parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev tap3 parent 1:11 handle 30: pfifo limit 5
tc filter add dev tap3 protocol ip parent 1:0 prio 1 u32 match ip src
192.168.57.100 flowid 1:10
tc filter add dev tap3 protocol ip parent 1:0 prio 1 u32 match ip src
192.168.57.101 flowid 1:11
```



QEMU (ubuntutest)

```
root@ubuntutest:/home/openstack# netperf -H 192.168.57.102 -t UDP_STREAM
MIGRATED UDP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.57.102 () port 0 AF_INET : demo
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

212992   65507   14.77         243      0       8.62
212992           14.77           5               0.18

root@ubuntutest:/home/openstack#
```

QEMU (ubuntutest2)

```
root@ubuntutest2:/home/openstack# netperf -H 192.168.57.102 -t UDP_STREAM
MIGRATED UDP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.57.102 () port 0 AF_INET : demo
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

212992   65507   10.58         722      0      35.75
212992           10.58          18              0.89

root@ubuntutest2:/home/openstack# _
```

3.5 Libvirt: CPU

- virsh capabilities
  - 有两台机器，一台Desktop
  - virsh capabilities > desktop.xml

```
<capabilities>
 <host>
  <uuid>004a7ff2-faec-de11-81d0-b9f7</uuid>
  <cpu>
   <arch>x86_64</arch>
   <model>Westmere</model>
   <vendor>Intel</vendor>
   <topology sockets='1' cores='2' threads='2'/>
   <feature name='rdtscp'/>
   <feature name='pcid'/>
   <feature name='pdcm'/>
   <feature name='xtpr'/>
   <feature name='tm2'/>
   <feature name='est'/>
   <feature name='smx'/>
   <feature name='vmx'/>
   <feature name='ds_cpl'/>
   <feature name='monitor'/>
   <feature name='dtes64'/>
   <feature name='pclmuldq'/>
   <feature name='pbe'/>
   <feature name='tm'/>
   <feature name='ht'/>
   <feature name='ss'/>
   <feature name='acpi'/>
   <feature name='ds'/>
   <feature name='vme'/>
  </cpu>
  <power_management>
   <suspend_mem/>
   <suspend_disk/>
   <suspend_hybrid/>
  </power_management>
  <migration_features>
   <live/>
   <uri_transports>
    <uri_transport>tcp</uri_transport>
   </uri_transports>
  </migration_features>
  <topology>
   <cells num='1'>
    <cell id='0'>
     <memory unit='KiB'>3836492</memory>
     <cpus num='4'>
      <cpu id='0' socket_id='0' core_id='0' siblings='0-1'/>
      <cpu id='1' socket_id='0' core_id='0' siblings='0-1'/>
      <cpu id='2' socket_id='0' core_id='2' siblings='2-3'/>
      <cpu id='3' socket_id='0' core_id='2' siblings='2-3'/>
     </cpus>
    </cell>
   </cells>
  </topology>
  <secmodel>
   <model>none</model>
   <doi>0</doi>
  </secmodel>
  <secmodel>
   <model>dac</model>
   <doi>0</doi>
   <baselabel type='kvm'>+107:+112</baselabel>
   <baselabel type='qemu'>+107:+112</baselabel>
  </secmodel>
 </host>
```

3.5 Libvirt: CPU

```
<guest>
  <os_type>hvm</os_type>
  <arch name='x86_64'>
   <wordsize>64</wordsize>
   <emulator>/usr/bin/qemu-system-x86_64</emulator>
   <machine canonical='pc-i440fx-trusty' maxCpus='255'>pc</machine>
   <machine maxCpus='255'>pc-1.3</machine>
   <machine maxCpus='255'>pc-0.12</machine>
   <machine maxCpus='255'>pc-q35-1.6</machine>
   <machine maxCpus='255'>pc-q35-1.5</machine>
   <machine maxCpus='1'>xenpv</machine>
   <machine maxCpus='255'>pc-i440fx-1.6</machine>
   <machine maxCpus='255'>pc-i440fx-1.7</machine>
   <machine maxCpus='255'>pc-0.11</machine>
   <machine maxCpus='255'>pc-1.2</machine>
   <machine maxCpus='255'>pc-0.10</machine>
   <machine maxCpus='1'>isapc</machine>
   <machine maxCpus='255'>pc-q35-1.4</machine>
   <machine maxCpus='128'>xenfv</machine>
   <machine maxCpus='255'>pc-0.15</machine>
   <machine maxCpus='255'>pc-0.14</machine>
   <machine maxCpus='255'>pc-i440fx-1.5</machine>
   <machine maxCpus='255'>pc-i440fx-1.4</machine>
   <machine canonical='pc-q35-2.0' maxCpus='255'>q35</machine>
   <machine maxCpus='255'>pc-1.1</machine>
   <machine maxCpus='255'>pc-q35-1.7</machine>
   <machine maxCpus='255'>pc-1.0</machine>
   <machine maxCpus='255'>pc-i440fx-2.0</machine>
   <machine maxCpus='255'>pc-0.13</machine>
   <domain type='qemu'>
   </domain>
```

```
<domain type='kvm'>
      <emulator>/usr/bin/kvm-spice</emulator>
      <machine canonical='pc-i440fx-trusty' maxCpus='255'>pc</machine>
      <machine maxCpus='255'>pc-1.3</machine>
      <machine maxCpus='255'>pc-0.12</machine>
      <machine maxCpus='255'>pc-q35-1.6</machine>
      <machine maxCpus='255'>pc-q35-1.5</machine>
      <machine maxCpus='1'>xenpv</machine>
      <machine maxCpus='255'>pc-i440fx-1.6</machine>
      <machine maxCpus='255'>pc-i440fx-1.7</machine>
      <machine maxCpus='255'>pc-0.11</machine>
      <machine maxCpus='255'>pc-1.2</machine>
      <machine maxCpus='255'>pc-0.10</machine>
      <machine maxCpus='1'>isapc</machine>
      <machine maxCpus='255'>pc-q35-1.4</machine>
      <machine maxCpus='128'>xenfv</machine>
      <machine maxCpus='255'>pc-0.15</machine>
      <machine maxCpus='255'>pc-0.14</machine>
      <machine maxCpus='255'>pc-i440fx-1.5</machine>
      <machine maxCpus='255'>pc-i440fx-1.4</machine>
      <machine canonical='pc-q35-2.0' maxCpus='255'>q35</machine>
      <machine maxCpus='255'>pc-1.1</machine>
      <machine maxCpus='255'>pc-q35-1.7</machine>
      <machine maxCpus='255'>pc-1.0</machine>
      <machine maxCpus='255'>pc-i440fx-2.0</machine>
      <machine maxCpus='255'>pc-0.13</machine>
    </domain>
  </arch>
  <features>
   <cpuselection/>
   <deviceboot/>
   <acpi default='on' toggle='yes'/>
   <apic default='on' toggle='no'/>
  </features>
 </guest>
```

3.5 Libvirt: CPU

- 一台Server
  - virsh capabilities > server.xml

```
<capabilities>
 <host>
  <uuid>80590690-87d2-e311-b1b0-a0481cabdfb4</uuid>
  <cpu>
   <arch>x86_64</arch>
   <model>SandyBridge</model>
   <vendor>Intel</vendor>
   <topology sockets='1' cores='10' threads='2'/>
   <feature name='erms'/>
   <feature name='smep'/>
   <feature name='fsgsbase'/>
   <feature name='pdpe1gb'/>
   <feature name='rdrand'/>
   <feature name='f16c'/>
   <feature name='osxsave'/>
   <feature name='dca'/>
   <feature name='pcid'/>
   <feature name='pdcm'/>
   <feature name='xtpr'/>
   <feature name='tm2'/>
   <feature name='est'/>
   <feature name='smx'/>
   <feature name='vmx'/>
   <feature name='ds_cpl'/>
   <feature name='monitor'/>
   <feature name='dtes64'/>
   <feature name='pbe'/>
   <feature name='tm'/>
   <feature name='ht'/>
   <feature name='ss'/>
   <feature name='acpi'/>
   <feature name='ds'/>
   <feature name='vme'/>
  </cpu>
  <power_management><suspend_mem/><suspend_disk/><suspend_hybrid/></power_management><migration_features><live/><uri_transports><uri_transport>tcp</uri_transport></uri_transports></migration_features>
    <topology>
     <cells num='1'>
      <cell id='0'>
       <memory unit='KiB'>65904468</memory>
       <cpus num='20'>
        <cpu id='0' socket_id='0' core_id='0' siblings='0,10'/>
        <cpu id='1' socket_id='0' core_id='1' siblings='1,11'/>
        <cpu id='2' socket_id='0' core_id='2' siblings='2,12'/>
        <cpu id='3' socket_id='0' core_id='3' siblings='3,13'/>
        <cpu id='4' socket_id='0' core_id='4' siblings='4,14'/>
        <cpu id='5' socket_id='0' core_id='8' siblings='5,15'/>
        <cpu id='6' socket_id='0' core_id='9' siblings='6,16'/>
        <cpu id='7' socket_id='0' core_id='10' siblings='7,17'/>
        <cpu id='8' socket_id='0' core_id='11' siblings='8,18'/>
        <cpu id='9' socket_id='0' core_id='12' siblings='9,19'/>
        <cpu id='10' socket_id='0' core_id='0' siblings='0,10'/>
        <cpu id='11' socket_id='0' core_id='1' siblings='1,11'/>
        <cpu id='12' socket_id='0' core_id='2' siblings='2,12'/>
        <cpu id='13' socket_id='0' core_id='3' siblings='3,13'/>
        <cpu id='14' socket_id='0' core_id='4' siblings='4,14'/>
        <cpu id='15' socket_id='0' core_id='8' siblings='5,15'/>
        <cpu id='16' socket_id='0' core_id='9' siblings='6,16'/>
        <cpu id='17' socket_id='0' core_id='10' siblings='7,17'/>
        <cpu id='18' socket_id='0' core_id='11' siblings='8,18'/>
        <cpu id='19' socket_id='0' core_id='12' siblings='9,19'/>
       </cpus>
      </cell>
     </cells>
    </topology>
</host>
```

3.5 Libvirt: CPU

- 在Desktop上运行：virsh cpu-compare server.xml
  - CPU described in server.xml is incompatible with host CPU
- 在Server上运行：virsh cpu-compare desktop.xml
  - Host CPU is a superset of CPU described in desktop.xml
- 如果要在两个机器之间进行Migration，则需要列出共有的feature，让Guest机器使用共有的feature才可以
  - cat server.xml desktop.xml > both-desktop-server.xml
  - virsh cpu-baseline both-desktop-server.xml
  - 将结果copy到guest domain的xml里面
  - match=”minimum”表示Host至少应该包含guest xml里面的这些feature，如果host有更多的feature，也让guest使用
  - match=”exact”表示Host应该至少包含guest xml里面的feature，即便有多的feature，也不让guest使用
  - match=”strict”表示Host和guest必须有完全一样的feature

```
<cpu mode='custom' match='exact'>
  <model fallback='allow'>Westmere</model>
  <vendor>Intel</vendor>
  <feature policy='require' name='rdtscp'/>
  <feature policy='require' name='pcid'/>
  <feature policy='require' name='pdcm'/>
  <feature policy='require' name='xtpr'/>
  <feature policy='require' name='tm2'/>
  <feature policy='require' name='est'/>
  <feature policy='require' name='smx'/>
  <feature policy='require' name='vmx'/>
  <feature policy='require' name='ds_cpl'/>
  <feature policy='require' name='monitor'/>
  <feature policy='require' name='dtes64'/>
  <feature policy='require' name='pclmuldq'/>
  <feature policy='require' name='pbe'/>
  <feature policy='require' name='tm'/>
  <feature policy='require' name='ht'/>
  <feature policy='require' name='ss'/>
  <feature policy='require' name='acpi'/>
  <feature policy='require' name='ds'/>
  <feature policy='require' name='vme'/>
</cpu>
```

3.5 Libvirt: CPU

- CPU affinity
  - 查看CPU的架构
    - virsh nodeinfo
    - 一个CPU槽，两个核，超线程
  - 可以将VCPU绑定在指定的物理CPU上
    - <vcpus cpuset='4-7'>4</vcpus>
  - 查看VCPU和CPU之间的映射
    - virsh vcpuinfo ubuntutest

```
root@popsuper1982:/home/openstack# virsh nodeinfo
CPU model:           x86_64
CPU(s):              4
CPU frequency:       1199 MHz
CPU socket(s):       1
Core(s) per socket:  2
Thread(s) per core:  2
NUMA cell(s):        1
Memory size:         3836492 KiB
```

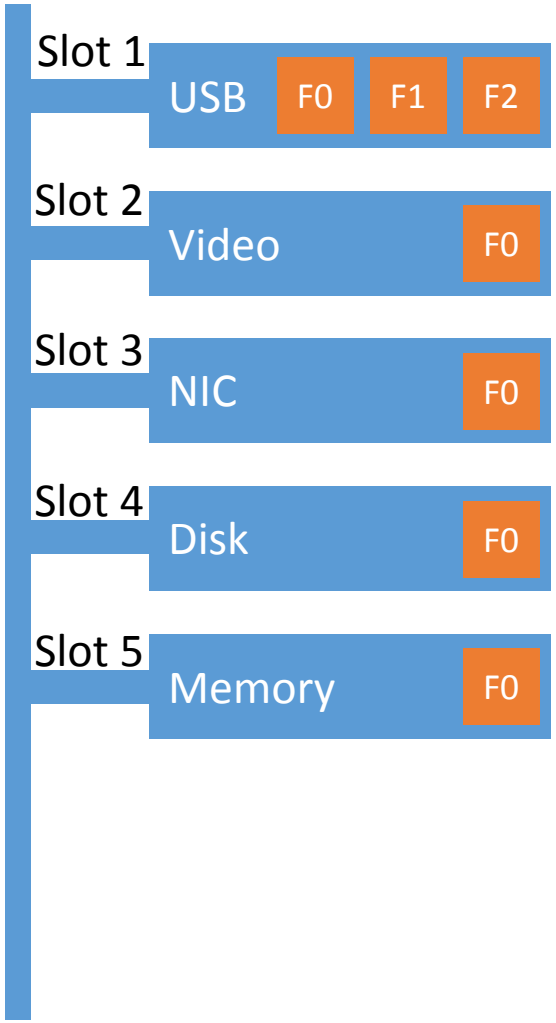
```
root@popsuper1982:/home/openstack# virsh vcpuinfo ubuntutest
VCPU:           0
CPU:            3
State:          running
CPU time:       49.7s
CPU Affinity:   yyyy
```

```
root@                                  # virsh vcpuinfo instance-0000000c
VCPU:           0
CPU:            5
State:          running
CPU time:       540979.1s
CPU Affinity:   yyyyyyyyyyyyyyyyyyyy

VCPU:           1
CPU:            7
State:          running
CPU time:       467152.0s
CPU Affinity:   yyyyyyyyyyyyyyyyyyyy

VCPU:           2
CPU:            11
State:          running
CPU time:       444465.4s
CPU Affinity:   yyyyyyyyyyyyyyyyyyyy

VCPU:           3
CPU:            9
State:          running
CPU time:       431381.7s
CPU Affinity:   yyyyyyyyyyyyyyyyyyyy
```

3.6 Libvirt: PCI

- PCI(Peripheral Component Interconnect)是设备总线标准
  - 一个PCI子系统包含四个模块
    - Bus
      - 最多256个Bus，常用的是Bus 0和Bus 1
    - Device
      - Device就是连接到Bus上的设备，如声卡，网卡等，最多32个
    - Function
      - 每个Device有至少有一个Function 0，最多8个
    - Register
      - 每个Function有256 eight-bit registers
  - virsh dumpxml ubuntutest查看里面的PCI配置，使得机器重启的时候，PCI配置不变，使得Guest对设备的访问处于稳定状态

3.6 Libvirt: PCI & virtio




```
<controller type='usb' index='0'>
  <alias name='usb0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'/>
</controller>
```

```
<video>
  <model type='cirrus' vram='9216' heads='1'/>
  <alias name='video0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
</video>
```

```
<interface type='bridge'>
  <mac address='fa:16:3e:6e:89:ce'/>
  <source bridge='br0'/>
  <target dev='tap1'/>
  <model type='virtio'/>
  <alias name='net0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

```
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2' cache='none'/>
  <source file='/home/openstack/images/ubuntutest1.qcow2'/>
  <target dev='vda' bus='virtio'/>
  <alias name='virtio-disk0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
</disk>
```

```
<memballoon model='virtio'>
  <alias name='balloon0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
</memballoon>
```

PCI Bus 0x00

3.7 Libvirt: libguestfs

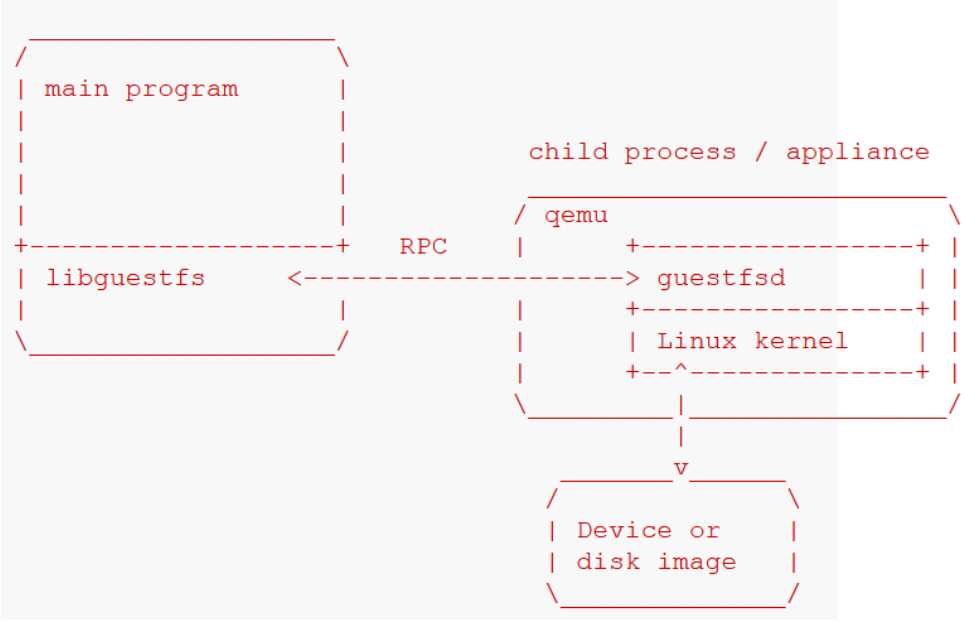
- Libguestfs可以在不启动虚拟机的情况下，编辑Image
- 安装： apt-get install libguestfs-tools
- 编辑一个Image：
  - guestfish -a trusty-server-cloudimg-amd64-disk1.img
  - 接着运行run，则一个虚拟机启动了
  - 查看所有的文件系统
    - list-filesystems
  - Mount这个文件系统
    - mount /dev/sda1 /

```
root@popsuper1982:/home/openstack/images# guestfish -a trusty-server-cloudimg-amd64-disk1.img

Welcome to guestfish, the guest filesystem shell for
editing virtual machine filesystems and disk images.

Type: 'help' for help on commands
      'man' to read the manual
      'quit' to quit the shell

><fs> run
><fs> list-filesystems
/dev/sda1: ext4
><fs> mount /dev/sda1 /
><fs> ls /
bin
boot
dev
etc
home
initrd.img
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
```

```
root@popsuper1982:/home/openstack# ps aux | grep qemu
root     15272  5.1  2.3 1912092 91444 pts/6   Sl   15:15   0:02 /usr/bin/qemu-system-x86_64 -global virtio-blk-pci.scsi=off -nodefconfig -enable-fips -nodefaults -nographi
c -machine accel=kvm:tcg -m 500 -no-reboot -rtc driftfix=slew -no-hpet -no-kvm-pit-reinjection -kernel /var/tmp/.guestfs-0/kernel.15263 -initrd /var/tmp/.guestfs-0/initrd.1
5263 -device virtio-scsi-pci,id=scsi -drive file=trusty-server-cloudimg-amd64-disk1.img,cache=writeback,id=hd0,if=none -device scsi-hd,drive=hd0 -drive file=/var/tmp/.guest
fs-0/root.15263,snapshot=on,id=appliance,cache=unsafe,if=none -device scsi-hd,drive=appliance -device virtio-serial-pci -serial stdio -device sga -chardev socket,path=/tmp/
libguestfsFx3SXZ/guestfsd.sock,id=channel0 -device virtserialport,chardev=channel0,name=org.libguestfs.channel.0 -append panic=1 console=ttyS0 udevtimeout=600 no_timer_chec
k acpi=off printk.time=1 cgroup_disable=memory root=/dev/sdb selinux=0 TERM=linux
```

3.7 Libvirt: libguestfs

- libguestfs的架构
  - guestfish -a trusty-server-cloudimg-amd64-disk1.img启动的进程，也即那个交互命令行是main program
  - 运行run的时候，会创建一个child process，在child process中，qemu运行一个称为appliance的小的虚拟机。创建子进程是由guestfs_launch函数完成的
  - 在appliance中，运行了linux kernel和一系列用户空间的工具(LVM, ext2等)，以及一个后台进程guestfsd
  - main process中的libguestfs和这个guestfd通过RPC进行交互。
  - 由child process的kernel来操作disk image



3.7 Libvirt: libguestfs

- libguestfs是一个C的library，你可以写一个C的程序，将这个类库加载进去，调用它的API
  - 文档http://libguestfs.org/guestfs.3.html就描述了这些C的API
- 而guestfish是一个交互命令行，可以通过执行命令，他来调用C类库的API，帮我们完成操作
  - 文档http://libguestfs.org/guestfish.1.html描述了这些命令，几乎所有的API，都有对应的命令

3.7 Libvirt: libguestfs

- Libguestfs appliance的启动过程
  - 如果我们想看这个appliance启动的详细过程，则需要export LIBGUESTFS_DEBUG=1
  - 然后运行guestfish -a trusty-server-cloudimg-amd64-disk1.img
  - 然后运行run，打印出很多的东西
  - (1) 启动guestfish

```
libguestfs: launch: program=guestfish
libguestfs: launch: version=1.24.5
libguestfs: launch: backend registered: unix
libguestfs: launch: backend registered: uml
libguestfs: launch: backend registered: libvirt
libguestfs: launch: backend registered: direct
libguestfs: launch: backend=direct
libguestfs: launch: tmpdir=/tmp/libguestfsfu3rNX
libguestfs: launch: umask=0022
libguestfs: launch: euid=0
```

  - (2) 运行supermin

```
libguestfs: command: run: /usr/bin/supermin-helper
libguestfs: command: run: \ --verbose
libguestfs: command: run: \ -f checksum
libguestfs: command: run: \ --host-cpu x86_64
libguestfs: command: run: \ /usr/lib/guestfs/supermin.d
supermin helper [00000ms] whitelist = (not specified)
supermin helper [00000ms] host_cpu = x86_64
supermin helper [00000ms] dtb_wildcard = (not specified)
supermin helper [00000ms] inputs:
supermin helper [00000ms] inputs[0] = /usr/lib/guestfs/supermin.d
supermin helper [00000ms] outputs:
supermin helper [00000ms] kernel = (none)
supermin helper [00000ms] dtb = (none)
supermin helper [00000ms] initrd = (none)
supermin helper [00000ms] appliance = (none)
```

3.7 Libvirt: libguestfs

- (3) 选择kernel

```
checking modpath /lib/modules/3.13.0-24-generic is a directory
picked kernel vmlinuz-3.13.0-24-generic
supermin helper [00000ms] finished creating kernel
```

- (4) 选择initrd, root images, 创建appliance

```
supermin helper [00000ms] visiting /usr/lib/guestfs/supermin.d
supermin helper [00000ms] visiting /usr/lib/guestfs/supermin.d/base.img
supermin helper [00000ms] visiting /usr/lib/guestfs/supermin.d/daemon.img.gz
supermin helper [00000ms] visiting /usr/lib/guestfs/supermin.d/hostfiles
supermin helper [00019ms] visiting /usr/lib/guestfs/supermin.d/init.img
supermin helper [00019ms] visiting /usr/lib/guestfs/supermin.d/udev-rules.img
supermin helper [00019ms] adding kernel modules
supermin helper [00067ms] finished creating appliance
libguestfs: checksum of existing appliance: d47e235f08edcbee72d2e82f8610e052dc7f90c26ce60b50abe02dc5677b413d
```

- (5) 检测qemu

```
libguestfs: [00071ms] begin testing qemu features
libguestfs: command: run: /usr/bin/qemu-system-x86_64
libguestfs: command: run: \ -nographic
libguestfs: command: run: \ -help
libguestfs: command: run: /usr/bin/qemu-system-x86_64
libguestfs: command: run: \ -nographic
libguestfs: command: run: \ -version
libguestfs: qemu version 2.0
libguestfs: command: run: /usr/bin/qemu-system-x86_64
libguestfs: command: run: \ -nographic
libguestfs: command: run: \ -machine accel=kvm:tcg
libguestfs: command: run: \ -device ?
libguestfs: [00146ms] finished testing qemu features
```

3.7 Libvirt: libguestfs

- (6) 启动qemu appliance

```
[00147ms] /usr/bin/qemu-system-x86_64 \
    -global virtio-blk-pci.scsi=off \
    -nodefconfig \
    -enable-fips \
    -nodefaults \
    -nographic \
    -machine accel=kvm:tcg \
    -m 500 \
    -no-reboot \
    -rtc driftfix=slew \
    -no-hpet \
    -no-kvm-pit-reinjection \
    -kernel /var/tmp/.guestfs-0/kernel.15321 \
    -initrd /var/tmp/.guestfs-0/initrd.15321 \
    -device virtio-scsi-pci,id=scsi \
    -drive file=trusty-server-cloudimg-amd64-disk1.img,cache=writeback,id=hd0,if=none \
    -device scsi-hd,drive=hd0 \
    -drive file=/var/tmp/.guestfs-0/root.15321,snapshot=on,id=appliance,cache=unsafe,if=none \
    -device scsi-hd,drive=appliance \
    -device virtio-serial-pci \
    -serial stdio \
    -device sga \
    -chardev socket,path=/tmp/libguestfsfu3rNX/guestfsd.sock,id=channel0 \
    -device virtserialport,chardev=channel0,name=org.libguestfs.channel.0 \
    -append 'panic=1 console=ttyS0 udevtimeout=600 no_timer_check acpi=off printk.time=1 cgroup_disable=memory root=/dev/sdb selinux=0 guestfs_verbose=1 TERM=linux'
```

appliance缓存在/var/tmp/.guestfs-<UID>

image是第一个disk

第二个是appliance的disk

3.7 Libvirt: libguestfs

- (7)启动initrd

```
supermin: mounting /proc
supermin: uptime: 0.51 0.16
supermin: ext2 mini initrd starting up: 4.1.6 zlib xz
supermin: cmdline: panic=1 console=ttyS0 udevtimeout=600 no_timer_check acpi=off printk.time=1 cgroup_disable=memory root=/dev/sdb selinux=0 guestfs_verbose=1 TERM=linux
```

- (8) load kernel modules

```
supermin: mounting /sys
supermin: internal insmod megaraid_mm.ko
[    0.517884] megaraid cmm: 2.20.2.7 (Release Date: Sun Jul 16 00:01:03 EST 2006)
supermin: internal insmod megaraid_mbox.ko
[    0.520678] megaraid: 2.20.5.1 (Release Date: Thu Nov 16 15:32:35 EST 2006)
supermin: internal insmod megaraid_sas.ko
[    0.524201] megasas: 06.700.06.00-rc1 Sat. Aug. 31 17:00:00 PDT 2013
supermin: internal insmod megaraid.ko
supermin: internal insmod libcrc32c.ko
supermin: internal insmod crc32-pclmul.ko
[    0.529605] PCLMULQDQ-NI instructions are not detected.
insmod: init_module: crc32-pclmul.ko: No such device
supermin: internal insmod crct10dif-pclmul.ko
insmod: init_module: crct10dif-pclmul.ko: No such device
supermin: internal insmod crc-itu-t.ko
supermin: internal insmod crc32.ko
supermin: internal insmod crc-ccitt.ko
supermin: internal insmod crc7.ko
supermin: internal insmod crc8.ko
supermin: internal insmod sym53c8xx.ko
supermin: internal insmod ideapad_slidebar.ko
[    0.543587] ideapad_slidebar: DMI does not match
insmod: init_module: ideapad_slidebar.ko: No such device
supermin: internal insmod sparse-keymap.ko
supermin: internal insmod ideapad-laptop.ko
supermin: internal insmod virtio-rng.ko
supermin: internal insmod virtio_scsi.ko
```

3.7 Libvirt: libguestfs

- (9) mount sda, sdb

```
[    0.551444] scsi2 : Virtio SCSI HBA
[    0.553728] scsi 2:0:0:0: Direct-Access     QEMU     QEMU HARDDISK    2.0. PQ: 0 ANSI: 5
[    0.555115] scsi 2:0:1:0: Direct-Access     QEMU     QEMU HARDDISK    2.0. PQ: 0 ANSI: 5
[    0.581792] sd 2:0:0:0: [sda] 4612096 512-byte logical blocks: (2.36 GB/2.19 GiB)
[    0.583092] sd 2:0:0:0: Attached scsi generic sg0 type 0
[    0.584256] sd 2:0:0:0: [sda] Write Protect is off
[    0.585208] sd 2:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
[    0.586808] sd 2:0:1:0: Attached scsi generic sg1 type 0
[    0.587699] sd 2:0:1:0: [sdb] 8388608 512-byte logical blocks: (4.29 GB/4.00 GiB)
[    0.589350] sd 2:0:1:0: [sdb] Write Protect is off
[    0.590307] sd 2:0:1:0: [sdb] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
[    0.592305]  sda: sda1
[    0.593401] sd 2:0:0:0: [sda] Attached SCSI disk
[    0.594479]  sdb: unknown partition table
[    0.595595] sd 2:0:1:0: [sdb] Attached SCSI disk
```

- (10) 将sdb作为root device

```
supermin: picked /sys/block/sdb/dev as root device
supermin: creating /dev/root as block special 8:16
supermin: mounting new root on /root
[    0.598257] EXT4-fs (sdb): mounting ext2 file system using the ext4 subsystem
[    0.601400] EXT4-fs (sdb): mounted filesystem without journal. Opts:
```

- (11) 运行init

```
supermin: chroot
Starting /init script ...
[    0.665989] systemd-udevd[83]: starting version 204
[    0.699480] ACPI Exception: AE_BAD_PARAMETER, Thread 494880720 could not acquire Mutex [0x1] (20131115/utmutex-285)
[    0.700876] piix4_smbus 0000:00:01.3: SMBus Host Controller at 0xb100, revision 0
[    0.743024] device-mapper: multipath: version 1.6.0 loaded
/init: 63: /init: systemd-tmpfiles: not found
/init: 69: /init: cannot create /sys/block/{h,s,ub,v}d*/queue/scheduler: Directory nonexistent
[    1.787852] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i8042/serio1/input/input3
Cannot find device "eth0"
Cannot find device "eth0"
RTNETLINK answers: Network is unreachable
/init: 86: /init: mdadm: not found
/init: 90: /init: lvmetad: not found
```

3.7 Libvirt: libguestfs

- (12) 启动guestfsd

```
Module                  Size  Used by
kvm_intel             143060  0
kvm                   451511  1 kvm_intel
psmouse               102222  0
dm_multipath           22873  0
mac_hid                13205  0
scsi_dh                14882  1 dm_multipath
serio_raw              13462  0
i2c_piix4              22155  0
virtio_scsi            18330  1
virtio_rng             13135  0
ideapad_laptop         18216  0
sparse_keymap          13948  1 ideapad_laptop
sym53c8xx              76731  0
crc8                   12893  0
crc7                   12703  0
crc_ccitt              12707  0
crc32                  12714  0
crc_itu_t              12707  0
libcrc32c              12644  0
megaraid               44120  0
megaraid_sas           91199  0
megaraid_mbox          40158  0
megaraid_mm            18253  1 megaraid_mbox
Mon Oct 20 22:33:32 UTC 2014
clocksource: kvm-clock
uptime: 2.15 1.06
verbose daemon enabled
linux commmand line: panic=1 console=ttyS0 udevtimeout=600 no_timer_check acpi=off printk.time=1 cgroup_disable=memory root=/dev/sdb selinux=0 guestfs_verbose=1 TERM=linux
```

- (13) 开通一个端口，C类库会通过RPC连接这个端口

```
trying to open virtio-serial channel '/dev/virtio-ports/org.libguestfs.channel.0'
udevadm settle
libguestfs: recv_from_daemon: received GUESTFS_LAUNCH_FLAG
libguestfs: [03741ms] appliance is up
><fs>
```

3.7 Libvirt: libguestfs

```
><fs> add-drive /home/openstack/images/ubuntutest.img format:qcow2
><fs> run
><fs> list-devices
/dev/sda
/dev/sdb
><fs> list-partitions
/dev/sda1
/dev/sdb1
/dev/sdb2
/dev/sdb5
><fs> list-filesystems
/dev/sda1: ext4
/dev/sdb1: ext4
/dev/sdb2: unknown
/dev/sdb5: swap
```

- Guestfish的命令
  - 添加一个drive
    - 这个命令只有在run之前起作用
    - 对应的API是guestfs_add_drive_opts
    - add-drive filename [readonly:true|false] [format:..] [iface:..] [name:..] [label:..] [protocol:..] [server:..]
    - guestfish -a trusty-server-cloudimg-amd64-disk1.img，这个Image是第一个drive
    - add-drive /home/openstack/images/ubuntutest.img format:qcow2，添加一个drive
    - 运行run
    - 查看所有的device: list-devices
    - 查看所有的分区: list-partitions
    - 查看所有的文件系统: list-filesystems

```
root@popsuper1982:/home/openstack# ps aux | grep qemu
root     15757  0.5  2.3 1653480 92060 pts/3   Sl   01:02   0:02 /usr/bin/qemu-system-x86_64 -global virtio-blk-pci.scsi=off -nodefconfig -enable-fips -nodefaults -nographi
c -machine accel=kvm:tcg -m 500 -no-reboot -rtc driftfix=slew -no-hpet -no-kvm-pit-reinjection -kernel /var/tmp/.guestfs-0/kernel.15748 -initrd /var/tmp/.guestfs-0/initrd.1
5748 -device virtio-scsi-pci,id=scsi -drive file=trusty-server-cloudimg-amd64-disk1.img,cache=writeback,id=hd0,if=none -device scsi-hd,drive=hd0 -drive file=/home/openstack
/images/ubuntutest.img,cache=writeback,format=qcow2,id=hd1,if=none -device scsi-hd,drive=hd1 -drive file=/var/tmp/.guestfs-0/root.15748,snapshot=on,id=appliance,cache=unsaf
e,if=none -device scsi-hd,drive=appliance -device virtio-serial-pci -serial stdio -device sga -chardev socket,path=/tmp/libguestfs4k7roA/guestfsd.sock,id=channel0 -device v
irtserialport,chardev=channel0,name=org.libguestfs.channel.0 -append panic=1 console=ttyS0 udevtimeout=600 no_timer_check acpi=off printk.time=1 cgroup_disable=memory root=
/dev/sdc selinux=0 TERM=linux
```

3.7 Libvirt: libguestfs

- Guestfish的命令
  - Mount文件系统
    - 对应的API是guestfs_mount
    - mount /dev/sda1 /
  - 文件系统操作
    - ls /
    - mkdir /mnt/sdb
    - mount /dev/sdb1 /mnt/sdb
    - ls /mnt/sdb
    - ls /mnt/sdb/home/openstack
    - cat /mnt/sdb/home/openstack/.bash_history
    - 更多文件系统命令chown，chmod，cp等都支持

```
><fs> mount /dev/sda1 /
><fs> ls /
bin
boot
dev
etc
home
initrd.img
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
```

```
><fs> ls /mnt
><fs> mkdir /mnt/sdb
><fs> mount /dev/sdb1 /mnt/sdb
><fs> ls /mnt/sdb
bin
boot
dev
etc
home
initrd.img
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
```

```
><fs> ls /mnt/sdb/home/openstack
.bash_history
.bash_logout
.bashrc
.cache
.profile
><fs> cat /mnt/sdb/home/openstack/.bash_history
sudo su
```

3.7 Libvirt: libguestfs

```
><fs> part-list /dev/sdb
[0] = {
  part_num: 1
  part_start: 1048576
  part_end: 3221225471
  part_size: 3220176896
}
[1] = {
  part_num: 2
  part_start: 3222273024
  part_end: 5367660543
  part_size: 2145387520
}
[2] = {
  part_num: 5
  part_start: 3222274048
  part_end: 5367660543
  part_size: 2145386496
}
```

```
><fs> part-get-bootable /dev/sdb 1
true
><fs> part-get-bootable /dev/sdb 2
false
```

- Guestfish的命令
  - 对partition的操作
    - part-list /dev/sdb
    - part-get-bootable /dev/sdb 1
    - 有个partition的命令包括：
      - part-add，part-del，part-disk，part-get-bootable，part-get-gpt-type，part-get-mbr-id，part-get-name，part-get-parttype，part-init，part-list，part-set-bootable，part-set-gpt-type，part-set-mbr-id，part-set-name，part-to-dev，part-to-partnum
  - 对LVM的操作
    - guestfish -a trusty-server-cloudimg-amd64-disk1.img
    - add-drive ./centos-5.8.new.qcow2 format:qcow2
    - run
    - 查看所有的PV: pvs-full
    - 查看所有的VG: vgs-full
    - 命令包含： lvcreate, lvcreate-free, lvm-canonical-lv-name, lvm-clear-filter, lvm-remove-all, lvm-set-filter, lvremove, lvrename, lvresize, lvresize-free, lvs, lvs-full, lvuuid, pvcreate, pvremove, pvresize, pvresize-size, pvs, pvs-full, pvuuid, vg-activate, vg-activate-all, vgchange-uuid, vgchange-uuid-all, vgcreate, vglvuuids, vgmeta, vgpvuuids, vgremove, vgrename, vgs, vgs-full, vgscan, vguuid

```
><fs> pvs-full
[0] = {
  pv_name: /dev/sdb2
  pv_uuid: K0B8RqHB21lvo7Eg3Db0d9Ya1Iw7C7AS
  pv_fmt: lvm2
  pv_size: 10603200512
  dev_size: 10627061760
  pv_free: 0
  pv_used: 10603200512
  pv_attr: ax-
  pv_pe_count: 316
  pv_pe_alloc_count: 316
  pv_tags:
  pe_start: 196608
  pv_mda_count: 1
  pv_mda_free: 94208
}
```

3.7 Libvirt: libguestfs

- Guestfish的命令
  - 下载和上传文件
    - guestfish -a ubuntutest.img
    - run
    - list-filesystems
    - mount /dev/sda1 /
    - download /home/openstack/.bash_history testdownload
    - upload instance01.xml /home/openstack/testupload

```
root@popsuper1982:/home/openstack/images# guestfish -a ubuntutest.img

Welcome to guestfish, the guest filesystem shell for
editing virtual machine filesystems and disk images.

Type: 'help' for help on commands
      'man' to read the manual
      'quit' to quit the shell

><fs> run
><fs> list-filesystems
/dev/sda1: ext4
/dev/sda2: unknown
/dev/sda5: swap
><fs> mount /dev/sda1 /
><fs> ls /home/openstack
.bash_history
.bash_logout
.bashrc
.cache
.profile
><fs> download /home/openstack/.bash_history testdownload
><fs> upload instance01.xml /home/openstack/testupload
><fs> ls /home/openstack/
.bash_history
.bash_logout
.bashrc
.cache
.profile
testupload
><fs> cat /home/openstack/testupload
<domain type='kvm'>
  <name>Instance01</name>
  <uuid>0f0806ab-531d-6134-5def-c5b495529211</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
```

3.7 Libvirt: libguestfs

- 所有的命令
  - https://rwmj.wordpress.com/2013/03/13/guestfish-now-supports-502-commands/

filesize	return the size of the file in bytes
filesystem-available	check if filesystem is available
fill	fill a file with octets
fill-dir	fill a directory with empty files
fill-pattern	fill a file with a repeating pattern of bytes
find	find all files and directories
find0	find all files and directories, returning NUL-separated list
findfs-label	find a filesystem by label
findfs-uuid	find a filesystem by UUID
fsck	run the filesystem checker
fstrim	trim free space in a filesystem
get-append	get the additional kernel options
get-attach-method	get the attach method
get-autosync	get autosync mode
get-cachedir	get the appliance cache directory
get-direct	get direct appliance mode flag
get-e2attrs	get ext2 file attributes of a file
get-e2generation	get ext2 file generation of a file
get-e2label	get the ext2/3/4 filesystem label
get-e2uuid	get the ext2/3/4 filesystem UUID
get-libvirt-requested-credential-challenge	challenge of i'th requested credential
get-libvirt-requested-credential-defresult	default result of i'th requested credential
get-libvirt-requested-credential-prompt	prompt of i'th requested credential
get-libvirt-requested-credentials	get list of credentials requested by libvirt
get-memsize	get memory allocated to the qemu subprocess
get-network	get enable network flag
get-path	get the search path
get-pgroup	get process group flag
get-pid	get PID of qemu subprocess
get-qemu	get the qemu binary
get-recovery-proc	get recovery process enabled flag
get-selinux	get SELinux enabled flag
get-smp	get number of virtual CPUs in appliance
get-tmpdir	get the temporary directory
get-trace	get command trace enabled flag
get-umask	get the current umask
get-verbose	get verbose mode
getcon	get SELinux security context
getxattr	get a single extended attribute
getxattrs	list extended attributes of a file or directory
glob	expand wildcards in command
glob-expand	expand a wildcard path
grep	return lines matching a pattern
grepi	return lines matching a pattern
grub-install	install GRUB 1
head	return first 10 lines of a file
head-n	return first N lines of a file
hexdump	dump a file in hexadecimal
hexedit	edit with a hex editor
hivex-close	close the current hivex handle
hivex-commit	commit (write) changes back to the hive
hivex-node-add-child	add a child node
hivex-node-children	return list of nodes which are subkeys of node
hivex-node-delete-child	delete a node (recursively)
hivex-node-get-child	return the named child of node
hivex-node-get-value	return the named value
hivex-node-name	return the name of the node
hivex-node-parent	return the parent of node
hivex-node-set-value	set or replace a single value in a node
hivex-node-values	return list of values attached to node
hivex-open	open a Windows Registry hive file
hivex-root	return the root node of the hive

I am an excel,
Double click me!

3.7 Libvirt: libguestfs

- Virt命令系列
  - 一个命令完成操作，无需启动交互命令行
  - Guestmount
    - 创建一个本地文件夹
      - mkdir testguestmount
    - 将image里面的/dev/sda1 mount到这个文件夹里面
      - guestmount -a ubuntutest.img -m /dev/sda1 testguestmount

```
root@popsuper1982:/home/openstack/images# mkdir testguestmount
root@popsuper1982:/home/openstack/images# guestmount -a ubuntutest.img -m /dev/sda1 testguestmount
root@popsuper1982:/home/openstack/images# cd testguestmount/
root@popsuper1982:/home/openstack/images/testguestmount# ls
bin  boot  dev  etc  home  initrd.img  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  vmlinuz
```

    - 结束编辑后
      - guestunmount testguestmount

3.7 Libvirt: libguestfs

- virt-builder
  - 可以快速的创建虚拟机镜像
  - update-guestfs-appliance
  - 查看所有的镜像类型
    - virt-builder –list
  - 创建一个Image
    - virt-builder fedora-20 -o myfedora.img --format qcow2 --size 20G
  - 设置root password, 放在文件里面
    - virt-builder fedora-20 --root-password file:/tmp/rootpw
  - 设置hostname
    - virt-builder fedora-20 --hostname virt.example.com
  - 安装软件
    - virt-builder fedora-20 –install “apache2“
  - 第一次启动运行脚本
    - virt-builder fedora-20 --firstboot /tmp/yum-update.sh

```
root@popsuper1982:/home/openstack/images# virt-builder --list
centos-6                 CentOS 6.5
centos-7.0               CentOS 7.0
cirros-0.3.1             CirrOS 0.3.1
debian-6                 Debian 6 (Squeeze)
debian-7                 Debian 7 (Wheezy)
fedora-18                Fedora?18
fedora-19                Fedora?19
fedora-20                Fedora?20
rhel-7rc                 Red Hat Enterprise Linux?7 Release Candidate
scientificlinux-6        Scientific Linux 6.5
ubuntu-10.04             Ubuntu 10.04 (Lucid)
ubuntu-12.04             Ubuntu 12.04 (Precise)
ubuntu-14.04             Ubuntu 14.04 (Trusty)
```

```
root@popsuper1982:/home/openstack/images# virt-builder fedora-20 -o myfedora.img --format qcow2 --size 20G
[   3.0] Downloading: http://libguestfs.org/download/builder/fedora-20.xz
######################################################################## 100.0%
[ 613.0] Creating disk image: myfedora.img
[ 614.0] Uncompressing: http://libguestfs.org/download/builder/fedora-20.xz
[ 644.0] Running virt-resize to expand the disk to 20.0G
[ 715.0] Opening the new disk
[ 720.0] Setting a random seed
[ 720.0] Random root password: yRGzRcY6ugtFbfa8 [did you mean to use --root-password?]
[ 720.0] Finishing off
Output: myfedora.img
Total usable space: 19.0G
Free space: 18.3G (96%)
root@popsuper1982:/home/openstack/images# qemu-img info myfedora.img
image: myfedora.img
file format: qcow2
virtual size: 20G (21474836480 bytes)
disk size: 822M
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

3.7 Libvirt: libguestfs

- virt-ls -a myfedora.img /root/
- virt-cat -a myfedora.img /root/.bash_profile
- virt-copy-in -a myfedora.img desktop.xml /root/
- virt-copy-out -a myfedora.img /root/.bash_profile ./
- virt-df -a myfedora.img

```
root@popsuper1982:/home/openstack/images# virt-df -a myfedora.img
Filesystem                          1K-blocks       Used  Available  Use%
myfedora.img:/dev/sda1                 487652      62103     395853   13%
myfedora.img:/dev/sda3               19390236     652460   17729972    4%
```

- virt-list-filesystems -a myfedora.img
- virt-list-partitions myfedora.img

```
root@popsuper1982:/home/openstack/images# virt-list-partitions myfedora.img
/dev/sda1
/dev/sda2
/dev/sda3
```

```
root@popsuper1982:/home/openstack/images# virt-ls -a myfedora.img /root/
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
```

```
root@popsuper1982:/home/openstack/images# virt-cat -a myfedora.img /root/.bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
```

```
root@popsuper1982:/home/openstack/images# virt-copy-in -a myfedora.img desktop.xml /root/
root@popsuper1982:/home/openstack/images# virt-ls -a myfedora.img /root/
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
desktop.xml
```

```
root@popsuper1982:/home/openstack/images# virt-copy-out -a myfedora.img /root/.bash_profile ./
root@popsuper1982:/home/openstack/images# cat .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
```

3.7 Libvirt: libguestfs

- virt-customize — customize virtual machines
- virt-diff — differences
- virt-edit — edit a file
- virt-format — erase and make blank disks
- virt-inspector — inspect VM images
- virt-log — display log files
- virt-make-fs — make a filesystem
- virt-rescue — rescue shell
- virt-resize — resize virtual machines
- virt-sparsify — make virtual machines sparse (thin-provisioned)
- virt-sysprep — unconfigure a virtual machine before cloning
- virt-tar — archive and upload files
- virt-tar-in — archive and upload files
- virt-tar-out — archive and download files
- virt-win-reg — export and merge Windows Registry keys

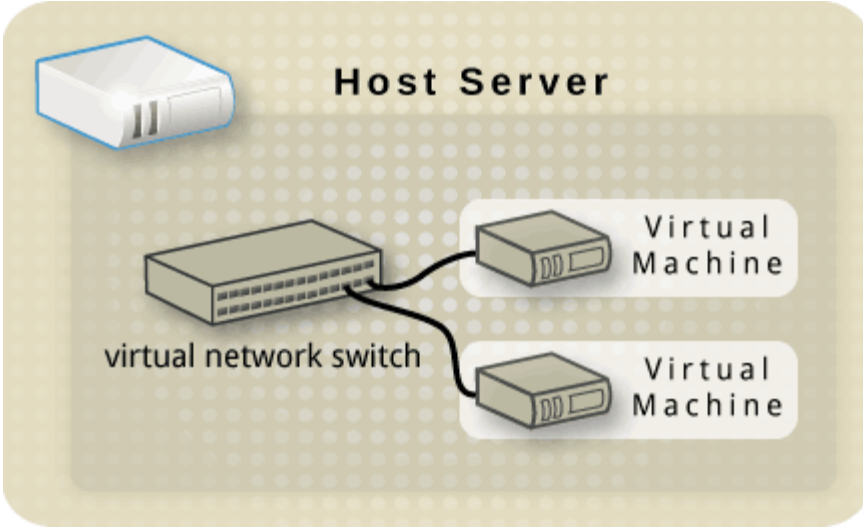
3.8 Libvirt: virtual networking

- Libvirt帮助管理虚拟网络
  - 默认网络default
  - 默认网桥virbr0
  - virsh net-list
  - virsh net-info default
  - virsh net-dumpxml default
  - 在domain中使用network

```
<interface type='network'>
    <source network='default'/>
</interface>
```

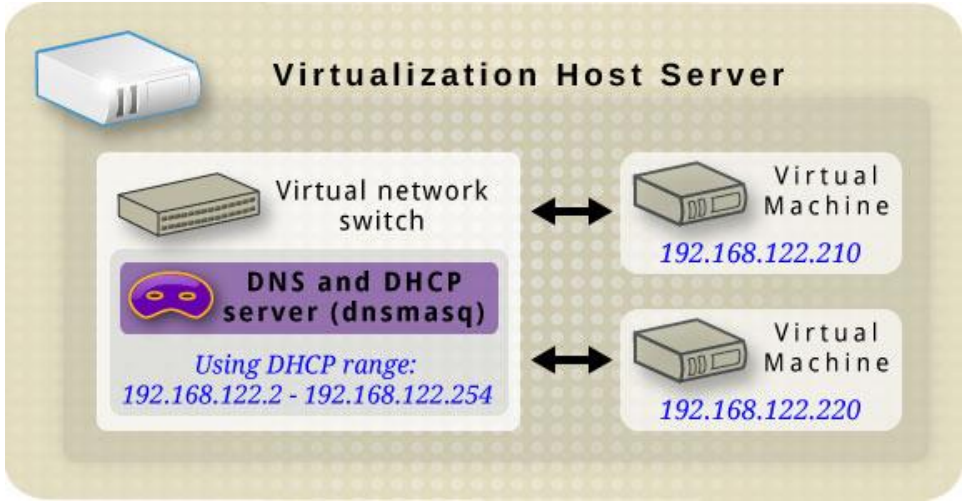
```
root@popsuper1982:/home/openstack/images# virsh net-list
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     yes           yes

root@popsuper1982:/home/openstack/images# virsh net-info default
Name:           default
UUID:           86891d0f-b6bb-49f8-a013-bae443c08d82
Active:         yes
Persistent:     yes
Autostart:      yes
Bridge:         virbr0

root@popsuper1982:/home/openstack/images# virsh net-dumpxml default
<network>
  <name>default</name>
  <uuid>86891d0f-b6bb-49f8-a013-bae443c08d82</uuid>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr0' stp='on' delay='0'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'/>
    </dhcp>
  </ip>
</network>
```

```
root@popsuper1982:/home/openstack/images# ip addr show virbr0
6: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether fe:54:00:e2:59:23 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
```



3.8 Libvirt: virtual networking

- DNS & DHCP
  - Libvirt使用dnsmasq作为DNS和DHCP Server




```
root@popsuper1982:/home/openstack/images# cat /var/lib/libvirt/dnsmasq/default.conf
##WARNING:  THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
##OVERWRITTEN AND LOST.  Changes to this configuration should be made using:
##    virsh net-edit default
## or other application using the libvirt API.
##
## dnsmasq conf file created by libvirt
strict-order
user=libvirt-dnsmasq
pid-file=/var/run/libvirt/network/default.pid
except-interface=lo
bind-dynamic
interface=virbr0
dhcp-range=192.168.122.2,192.168.122.254
dhcp-no-override
dhcp-leasefile=/var/lib/libvirt/dnsmasq/default.leases
dhcp-lease-max=253
dhcp-hostsfile=/var/lib/libvirt/dnsmasq/default.hostsfile
addn-hosts=/var/lib/libvirt/dnsmasq/default.addnhosts
```

```
root@popsuper1982:/home/openstack/images# ps aux | grep dnsmasq
libvirt+  1973  0.0  0.0  28208   768 ?        S    Sep10   0:03 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf
```

```
root@popsuper1982:/home/openstack/images# virsh list
 Id    Name                           State
----------------------------------------------------
 41    ubuntutest4                    running
```

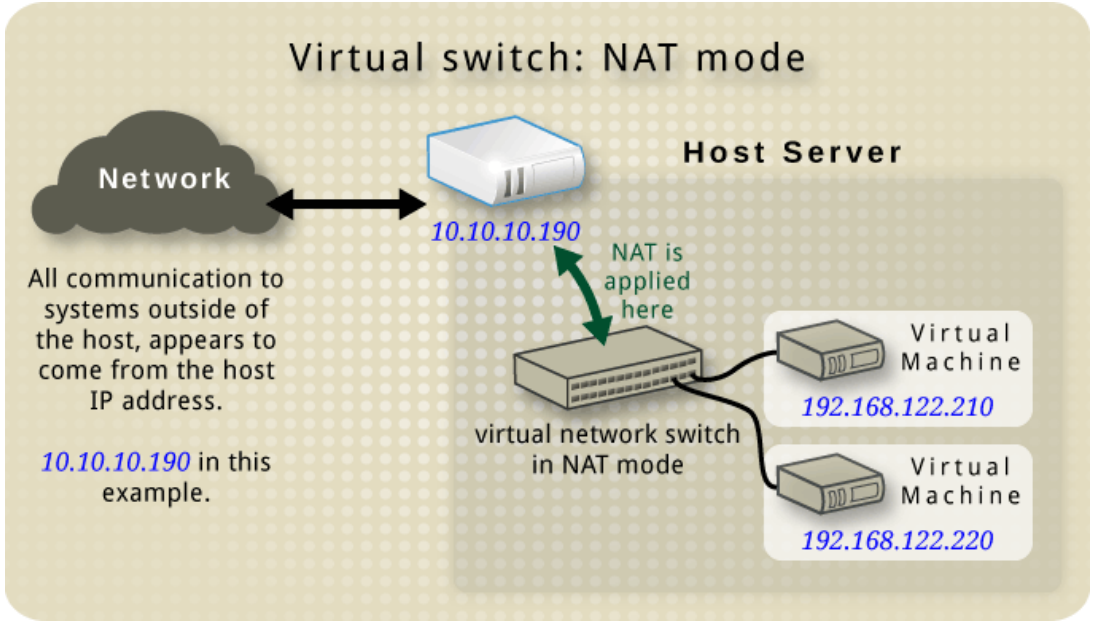
QEMU (ubuntutest4)

```
root@ubuntutest:/home/openstack# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:e2:59:23 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.17/24 brd 192.168.122.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fee2:5923/64 scope link
       valid_lft forever preferred_lft forever
```

3.8 Libvirt: virtual networking

- 默认的NAT模式
  - 由iptables实现
    - iptables -t nat -nvL

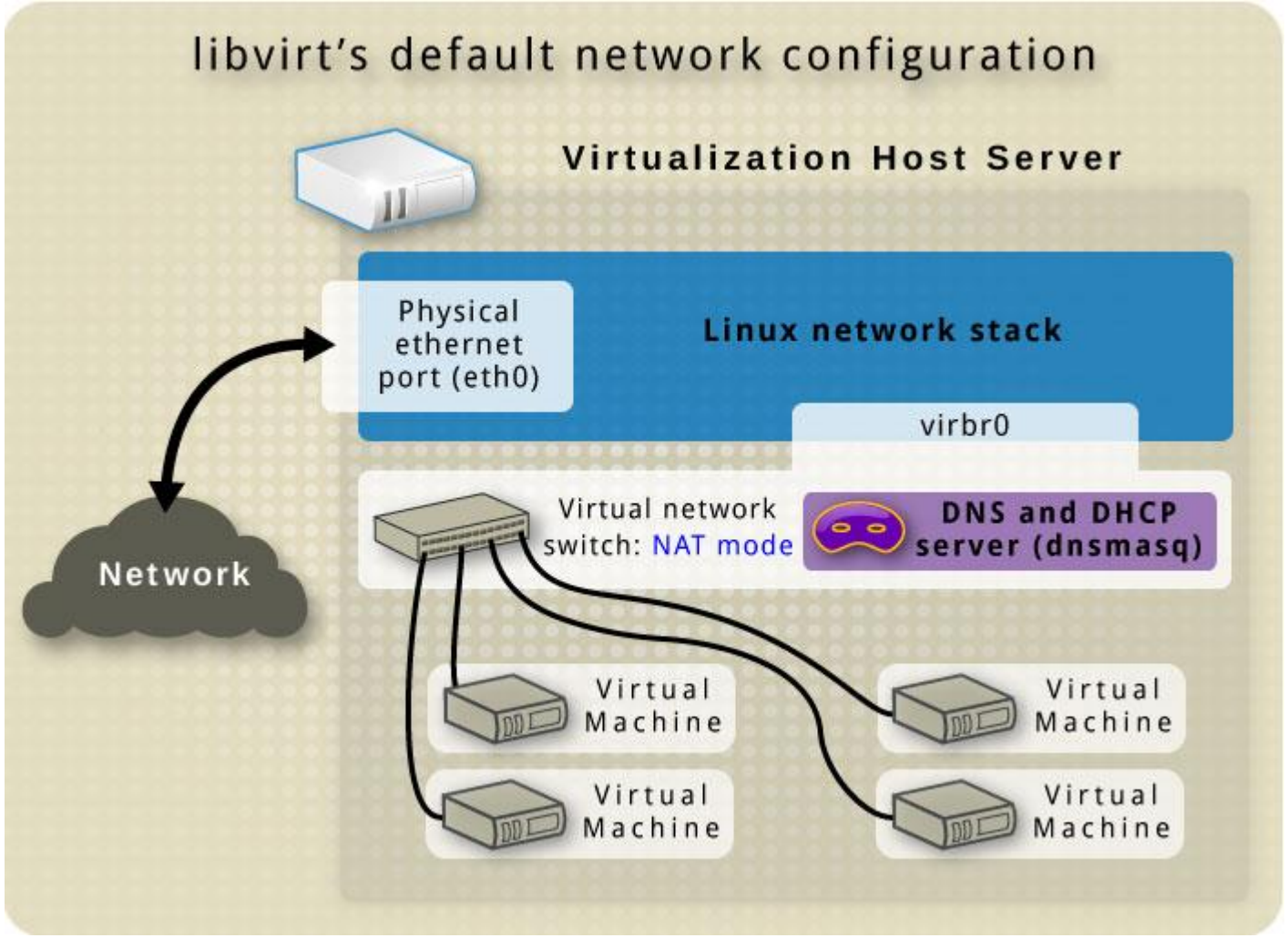
```
Chain POSTROUTING (policy ACCEPT 459 packets, 422K bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 RETURN     all  --  *      *       192.168.122.0/24     224.0.0.0/24
    0     0 RETURN     all  --  *      *       192.168.122.0/24     255.255.255.255
    0     0 MASQUERADE  tcp  --  *      *       192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-65535
    2   152 MASQUERADE  udp  --  *      *       192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-65535
    0     0 MASQUERADE  all  --  *      *       192.168.122.0/24    !192.168.122.0/24
```



MASQUERADE，地址伪装，在iptables中有着和snat相近的效果，但也有一些区别。
SNAT：出口ip的地址范围可以是一个，也可以是多个，但要求ip地址范围是确定的
MASQUERADE：如果出口ip地址不可以确定，不管现在eth0的出口获得了怎样的动态ip，MASQUERADE会自动读取eth0现在的ip地址然后做SNAT出去

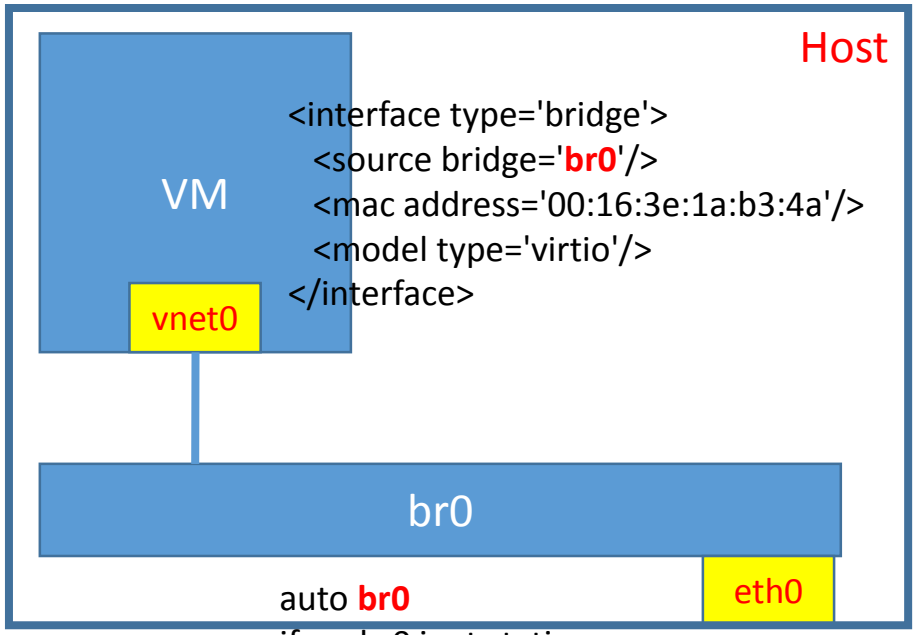
3.8 Libvirt: virtual networking

- 默认的配置



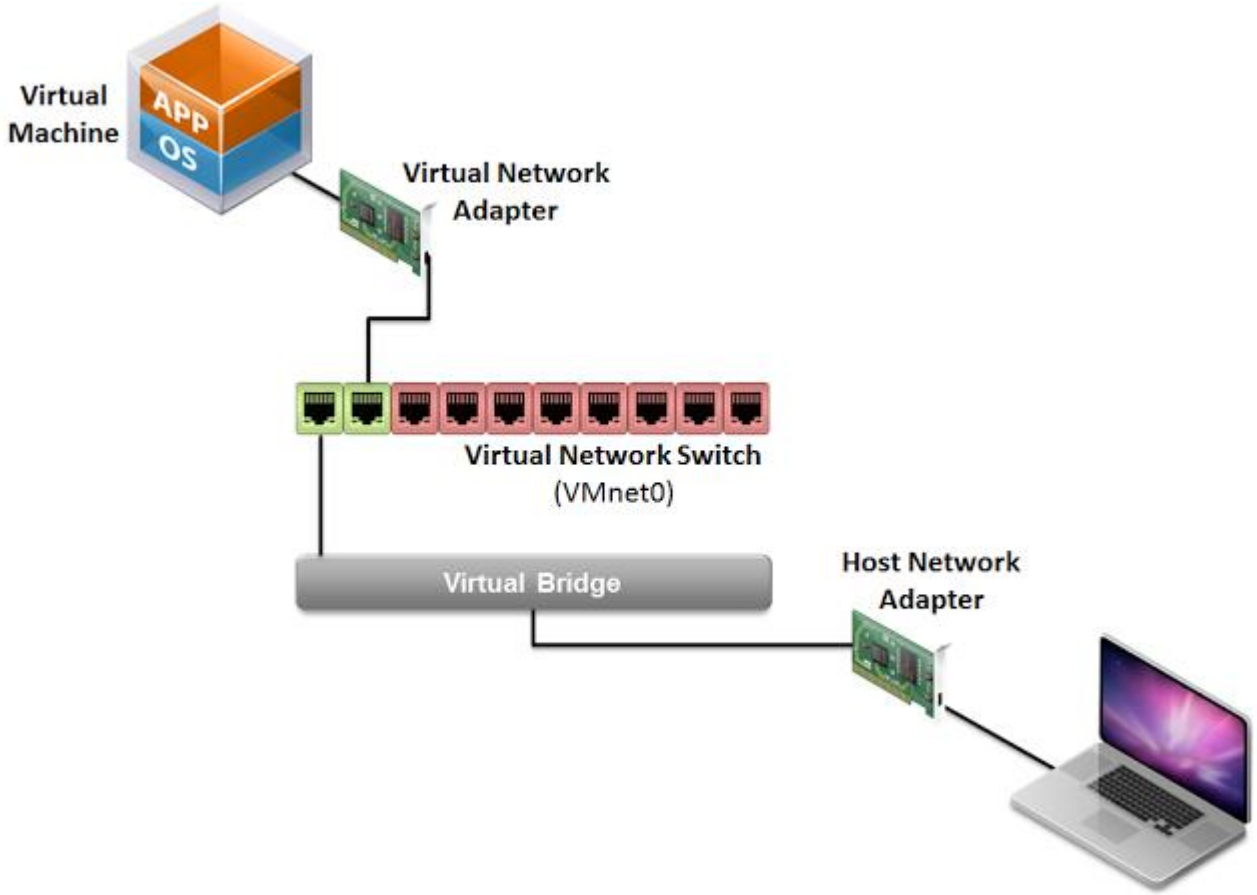
3.8 Libvirt: virtual networking

- Bridged Network




```
<interface type='bridge'>
  <source bridge='br0'/>
  <mac address='00:16:3e:1a:b3:4a'/>
  <model type='virtio'/>
</interface>
```

```
auto br0
iface br0 inet static
        address 192.168.2.4
        netmask 255.255.255.0
        network 192.168.2.0
        broadcast 192.168.2.255
        gateway 192.168.2.2
        bridge_ports eth0
        bridge_stp on
        bridge_maxwait 0
```



3.8 Libvirt: virtual networking

- Libvirt提供自己的network filter，基于ebtables, iptables, ip6tables
- 由下面的命令进行操作virsh：
  - nwfilter-define
  - nwfilter-dumpxml
  - nwfilter-edit
  - nwfilter-list
  - nwfilter-undefined
- 查看当前已经有的filter
  - virsh nwfilter-list

```
root@popsuper1982:/home/openstack# virsh nwfilter-list
 UUID                                   Name
--------------------------------------------------------------------
 43c53f7d-09b0-44d6-a20e-31d24d4d029c   allow-arp
 5cbaf802-e72d-440d-8c0c-f68afc9fda18   allow-dhcp
 3fcc9d6d-24c0-4367-be7a-284a9f2ea107   allow-dhcp-server
 c1fc2f42-2ea6-4fbe-801d-e3733c10d718   allow-incoming-ipv4
 85a8e9a3-d13b-4315-9e56-966e663eb37a   allow-ipv4
 7eea67d5-0cc0-455a-b797-99487155da4b   clean-traffic
 d9b34d1c-5420-4e16-98cf-1c75e3e2561f   no-arp-ip-spoofing
 d56047f1-ad4d-40f5-9532-f4104f03b84d   no-arp-mac-spoofing
 07903296-42ae-4cb8-bb7c-bd89b0297172   no-arp-spoofing
 8d87f453-7273-4087-b9e2-40d74573e75b   no-icmp
 802a1597-00c8-451c-aa4c-fd2f1be01126   no-ip-multicast
 d48b62fb-3f32-4d28-86da-59ece99f767c   no-ip-spoofing
 ef0e2247-f2d5-482f-810f-19b033774ba7   no-mac-broadcast
 56bd6e89-7c77-4503-ae01-437464465a8c   no-mac-spoofing
 466ad10b-6ba4-4b50-aa5b-32fa8f2142f6   no-other-l2-traffic
 1b7365aa-bf97-4e4a-ae6a-561ac025cef4   no-other-rarp-traffic
 81dafd32-6791-42ca-9e44-5f3b6cf23e5e   qemu-announce-self
 5eb99111-d9fc-4646-b528-774000ff6202   qemu-announce-self-rarp
```

- 这些filter也是通过xml进行定义的，我们可以查看详情
  - virsh nwfilter-dumpxml no-ip-multicast

```
root@popsuper1982:/home/openstack# virsh nwfilter-dumpxml no-ip-multicast
<filter name='no-ip-multicast' chain='ipv4' priority='-700'>
  <uuid>802a1597-00c8-451c-aa4c-fd2f1be01126</uuid>
  <rule action='drop' direction='out' priority='500'>
    <ip dstipaddr='224.0.0.0' dstipmask='4'/>
  </rule>
</filter>
```

3.8 Libvirt: virtual networking

- Network Filter XML
  - UUID
  - Name
  - Chain
    - 根为root chain，其他的chain都连接到root上，按照优先级依次访问
  - Rule表示规则，有下面的属性
    - Action：accept，drop
    - Direction：in, out, inout
    - priority
  - Rule支持多种protocol，可以设置参数
    - MAC (Ethernet)：root chain
    - VLAN (802.1Q): root/vlan
    - STP (Spanning Tree Protocol)
    - ARP/RARP: root or arp/rarp chain
    - IPv4: root or ipv4 chain
    - IPv6
    - TCP/UDP/SCTP: root
    - ICMP: root
    - IGMP, ESP, AH, UDPLITE, 'ALL': root
  - Filterref引用其他的filter

Chain (prefix)	Default priority
stp	-810
mac	-800
vlan	-750
ipv4	-700
ipv6	-600
arp	-500
rarp	-400

Attribute	Datatype	Semantics
srcmacaddr	MAC_ADDR	MAC address of sender
srcmacmask	MAC_MASK	Mask applied to MAC address of sender
dstmacaddr	MAC_ADDR	MAC address of destination
dstmacmask	MAC_MASK	Mask applied to MAC address of destination
protocolid	UINT16 (0x600-0xffff), STRING	Layer 3 protocol ID

Attribute	Datatype	Semantics
srcmacaddr	MAC_ADDR	MAC address of sender
srcmacmask	MAC_MASK	Mask applied to MAC address of sender
dstmacaddr	MAC_ADDR	MAC address of destination
dstmacmask	MAC_MASK	Mask applied to MAC address of destination
vlanid	UINT16 (0x0-0xfff, 0 - 4095)	VLAN ID
encap-protocol	UINT16 (0x03c-0xfff), String	Encapsulated layer 3 protocol ID

Attribute	Datatype	Semantics
srcmacaddr	MAC_ADDR	MAC address of sender
srcmacmask	MAC_MASK	Mask applied to MAC address of sender
dstmacaddr	MAC_ADDR	MAC address of destination
dstmacmask	MAC_MASK	Mask applied to MAC address of destination
srcipaddr	IP_ADDR	Source IP address
srcipmask	IP_MASK	Mask applied to source IP address
dstipaddr	IP_ADDR	Destination IP address
dstipmask	IP_MASK	Mask applied to destination IP address
protocol	UINT8, STRING	Layer 4 protocol identifier
srcportstart	UINT16	Start of range of valid source ports; requires protocol
srcportend	UINT16	End of range of valid source ports; requires protocol
dstportstart	UINT16	Start of range of valid destination ports; requires protocol
dstportend	UINT16	End of range of valid destination ports; requires protocol

3.8 Libvirt: virtual networking

- 实例：禁用ICMP
  - 启动一个虚拟机，网卡配置如下

```
<interface type='network'>
  <mac address='52:54:00:e2:59:23'/>
  <source network='default'/>
  <model type='rtl8139'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
</interface>
```

  - 在HOST上可以ping通虚拟机

```
root@popsuper1982:/home/openstack# ping 192.168.122.17
PING 192.168.122.17 (192.168.122.17) 56(84) bytes of data.
64 bytes from 192.168.122.17: icmp_seq=1 ttl=64 time=0.484 ms
64 bytes from 192.168.122.17: icmp_seq=2 ttl=64 time=0.424 ms
64 bytes from 192.168.122.17: icmp_seq=3 ttl=64 time=0.407 ms
^C
--- 192.168.122.17 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.407/0.438/0.484/0.037 ms
root@popsuper1982:/home/openstack#
```

```
QEMU (ubuntutest4)
openstack@ubuntutest:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNO
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_f
    link/ether 52:54:00:e2:59:23 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.17/24 brd 192.168.122.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fee2:5923/64 scope link
```

  - 定义一个filter
    - virsh nwfilter-define no-icmp.xml
    - virsh nwfilter-list

```
root@popsuper1982:/home/openstack# cat no-icmp.xml
<filter name='no-icmp'>
  <rule action='drop' direction='inout'>
    <icmp/>
  </rule>
</filter>
```

  - 将filter应用于虚拟机
  - 重启虚拟机
    - virsh destroy ubuntutest4
    - virsh start ubuntutest4

```
<interface type='network'>
  <mac address='52:54:00:e2:59:23'/>
  <source network='default'/>
  <model type='rtl8139'/>
  <filterref filter='no-icmp'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
</interface>
```

  - Ping不通了

3.9 Libvirt: Storage

- Storage Pool
  - 将存储设备作为一个池进行管理
  - 目录池dir：以主机的一个目录作为存储池，这个目录中包含文件的类型可以为各种虚拟机磁盘文件、镜像文件等。
  - 本地文件系统池fs：使用主机已经格式化好的块设备作为存储池，支持的文件系统类型包括 ext2,ext3,vfat 等。
  - 网络文件系统池netfs：使用远端网络文件系统服务器的导出目录作为存储池。默认为 NFS 网络文件系统。
  - 逻辑卷池logical：使用已经创建好的 LVM 卷组，或者提供一系列生成卷组的源设备，libvirt 会在其上创建卷组，生成存储池。
  - 磁盘卷池disk：使用磁盘作为存储池。
  - iSCSI 卷池：使用 iSCSI 设备作为存储池。

3.9 Libvirt: Storage

- 网络文件系统池netfs：
  - 目录池dir，本地文件系统池fs，网络文件系统池netfs三者很像，都是最终生成一个目录，这个目录里面的Image文件，可以作为虚拟机的硬盘，
    - 目录池：目录是管理员事先创建好的本地目录，Image就在这个目录下
    - 本地文件系统池：指定一个本地的文件系统，还没有mount，libvirt负责把这个文件系统mount到一个本地目录下，Image就在这个目录下
    - 网络文件系统池：指定一个网络文件系统，还没有mount，libvirt负责把这个网络文件系统mount到本地目录下，Image就在这个目录下
  - 服务器端安装nfs服务
    - apt-get install nfs-kernel-server
    - 在/etc/exports中添加/mnt/nfs　*(rw,sync,no_root_squash)
    - 重启服务service nfs-kernel-server restart

```
root@popsuper1982:/home/openstack# virsh pool-list
 Name                 State      Autostart
-------------------------------------------
 nfspool              active     no

root@popsuper1982:/home/openstack# virsh pool-info nfspool
Name:           nfspool
UUID:           58cfcbee-bc48-472e-94bf-df320cf1b0ba
State:          running
Persistent:     yes
Autostart:      no
Capacity:       171.19 GiB
Allocation:     150.79 GiB
Available:      20.40 GiB
```

3.9 Libvirt: Storage

- 网络文件系统池netfs：
  - 客户端
    - apt-get install nfs-common
    - 定义一个nfs-pool.xml
    - virsh pool-define nfs-pool.xml
    - virsh pool-start nfspool
    - 如果在pool-start之前，nfs目录里面已经有image文件，则自动添加为volume，之后添加的不会添加
    - virsh vol-list nfspool
    - 可以创建volume
      - virsh vol-create-as --pool nfspool --name nfsvol --capacity 2G
    - 可以删除volume
      - virsh vol-delete nfsvol --pool nfspool
    - 删除pool
      - virsh pool-destroy nfspool
      - virsh pool-undefine nfspool

```
<pool type="netfs">
  <name>nfspool</name>
   <source>
    <host name="16.158.166.197"/>
    <dir path="/mnt/nfs"/>
    <format type='nfs'/>
   </source>
   <target>
    <path>/home/openstack/nfs</path>
   </target>
</pool>
```

服务器地址

服务器目录

客户端目录

```
root@popsuper1982:/home/openstack# virsh vol-list nfspool
 Name                 Path
------------------------------------------------------------------
 nfsimage.img         /home/openstack/nfs/nfsimage.img
```

```
root@popsuper1982:/home/openstack# virsh vol-list nfspool
 Name                 Path
------------------------------------------------------------------
 nfsimage.img         /home/openstack/nfs/nfsimage.img
 nfsvol               /home/openstack/nfs/nfsvol

root@popsuper1982:/home/openstack# ls -l nfs/
total 8428
-rw-r--r-- 1 root root     197120 Oct 23 14:24 nfsimage.img
-rw------- 1 root root 2147483648 Oct 23  2014 nfsvol
```

3.9 Libvirt: Storage

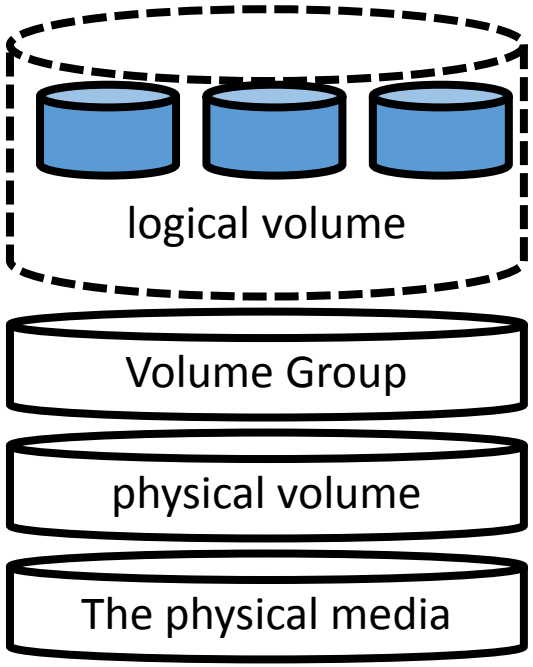
- 网络文件系统池netfs：
  - 虚拟机使用volume
    - virsh attach-disk ubuntutest4 --source /home/openstack/nfs/nfsvol --target vdb
    - virsh detach-disk ubuntutest4 --target vdb

```
root@popsuper1982:/home/openstack# virsh list
 Id    Name                           State
----------------------------------------------------
 48    ubuntutest4                    running

root@popsuper1982:/home/openstack# virsh domblklist ubuntutest4
Target     Source
------------------------------------------------
vda        /home/openstack/images/ubuntutest4.qcow2
```

```
root@popsuper1982:/home/openstack# virsh attach-disk ubuntutest4 --source /home/openstack/nfs/nfsvol  --target vdb
Disk attached successfully

root@popsuper1982:/home/openstack# virsh domblklist ubuntutest4
Target     Source
------------------------------------------------
vda        /home/openstack/images/ubuntutest4.qcow2
vdb        /home/openstack/nfs/nfsvol
```

3.9 Libvirt: Storage

- 逻辑卷池logical
  - 基于LVM的存储池
    - 安装LVM apt-get install lvm2
    - 创建PV
      - fdisk /dev/vdb



lvcreate -L 200M vg

vgcreate vg /dev/sdb

pvcreate /dev/vdb1

/dev/vdb1

```
root@ubuntutest:/home/openstack# fdisk /dev/vdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
Building a new DOS disklabel with disk identifier 0xda4aba84.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
  rtition number (1-4, default 1):
?sing default value 1
First sector (2048-20971519, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519):
Using default value 20971519

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 8e
Changed system type of partition 1 to 8e (Linux LVM)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@ubuntutest:/home/openstack# partprobe
```

```
Disk /dev/vdb: 10.7 GB, 10737418240 bytes
2 heads, 17 sectors/track, 616809 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xda4aba84

   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1            2048    20971519    10484736   8e  Linux LVM
```

3.9 Libvirt: Storage

- 逻辑卷池logical
  - 定义一个lvmpool.xml
    - virsh pool-define lvmpool.xml
    - virsh pool-start lvm_pool
    - virsh pool-list
    - 这个时候，我们能看到VG
  - 创建一个volume
    - virsh vol-create-as --pool lvm_pool --name vol3 --capacity 30M
    - virsh vol-list --pool lvm_pool

```
<pool type="logical">
  <name>lvm_pool</name>
  <source>
    <device path="/dev/vdb1"/>
  </source>
  <target>
   <path>/lvm_pool</path>
  </target>
</pool>
```

```
root@ubuntutest:/home/openstack# pvs
  PV         VG       Fmt  Attr PSize  PFree
  /dev/vdb1  lvm_pool lvm2 a--  10.00g 10.00g
```

```
root@ubuntutest:/home/openstack# virsh pool-list
 Name                 State      Autostart
-----------------------------------------
 lvm_pool             active     no

root@ubuntutest:/home/openstack# vgdisplay
  --- Volume group ---
  VG Name               lvm_pool
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               10.00 GiB
  PE Size               4.00 MiB
  Total PE              2559
  Alloc PE / Size       0 / 0
  Free  PE / Size       2559 / 10.00 GiB
  VG UUID               19sZU5-lBXT-46tg-FebA-pJG9-VZUi-ofdoB8
```

```
root@ubuntutest:/home/openstack# virsh vol-list --pool lvm_pool
 Name                 Path
------------------------------------------------------------------------------
 vol3                 /dev/lvm_pool/vol3

root@ubuntutest:/home/openstack# lvdisplay
  --- Logical volume ---
  LV Path                /dev/lvm_pool/vol3
  LV Name                vol3
  VG Name                lvm_pool
  LV UUID                9IaNs1-gMDk-T0zW-emLl-IbPR-Ca2z-O5XNfz
  LV Write Access        read/write
  LV Creation host, time ubuntutest, 2014-10-24 03:13:45 -0700
  LV Status              available
  # open                 0
  LV Size                32.00 MiB
  Current LE             8
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0
```

3.9 Libvirt: Storage

- iSCSI 卷池
  - 在服务器端
    - apt-get install iscsitarget
    - apt-get install tgt
    - /etc/default/iscsitarget中修改ISCSITARGET_ENABLE=true
    - service iscsitarget restart
    - 创建两个LVM所谓iscsi的backing storage
      - lvcreate -L 2G -n lvm_lun1 lvm_pool
      - lvcreate -L 3G -n lvm_lun2 lvm_pool
    - 创建一个iscsitarget
      - tgtadm --lld iscsi --op new --mode target --tid 2 -T iqn.2014-10.org.openstack:lvm_pool
    - 将Logic Volume加入刚才创建的iscsi target
      - tgtadm --lld iscsi --op new --mode logicalunit --tid 2 --lun 1 -b /dev/lvm_pool/lvm_lun1
      - tgtadm --lld iscsi --op new --mode logicalunit --tid 2 --lun 2 -b /dev/lvm_pool/lvm_lun2
      - tgtadm --lld iscsi --mode target --op show
    - 配置iscsi target监听链接
      - tgtadm --lld iscsi --op bind --mode target --tid 2 -I ALL

3.9 Libvirt: Storage

- iSCSI 卷池
  - 在客户端
    - 定义一个iscsi-pool.xml
    - virsh pool-define iscsi-pool.xml
    - virsh pool-start iscsipool
    - virsh pool-list
    - virsh vol-list --pool iscsipool

```
<pool type='iscsi'>
    <name>iscsipool</name>
    <source>
        <host name='192.168.122.17'/>
        <device path='iqn.2014-10.org.openstack:lvm_pool'/>
    </source>
    <target>
        <path>/dev/disk/by-path</path>
    </target>
</pool>
```

```
root@popsuper1982:/home/openstack# virsh pool-list
 Name                 State      Autostart
-----------------------------------------
 iscsipool            active     no

root@popsuper1982:/home/openstack# virsh vol-list --pool iscsipool
 Name                 Path
------------------------------------------------------------------------------
 unit:0:0:1           /dev/disk/by-path/ip-192.168.122.17:3260-iscsi-iqn.2014-10.org.openstack:lvm_pool-lun-1
 unit:0:0:2           /dev/disk/by-path/ip-192.168.122.17:3260-iscsi-iqn.2014-10.org.openstack:lvm_pool-lun-2
```

3.9 Libvirt: Storage

- Disk Encryption
  - 前面介绍qcow2的时候，它支持加密
  - qemu-img convert -o encryption -f qcow2 -O qcow2 ubuntutest4.qcow2 ubuntutest4-encrypt.qcow2
  - 当虚拟机启动的时候，需要在monitor中输入密码
  - 使用libvirt管理虚拟机，我们可不想每启动一个虚拟机后，都要登录monitor输入密码，libvirt帮助我们管理密码

3.9 Libvirt: Storage

- Disk Encryption
  - 定义一个secret disk-secret.xml
    - virsh secret-define disk-secret.xml
    - virsh secret-list
  - 设置密码
    - MYSECRET=`printf %s "123456" | base64`
    - virsh secret-set-value 027f6cca-e293-45aa-a4d1-6ddbd8b7f706 $MYSECRET
  - 创建一个disk pool secret-disk-pool.xml
    - virsh pool-define secret-disk-pool.xml
  - 创建一个disk volume secret-disk.xml
    - virsh vol-create SecretDiskPool secret-disk.xml

```
root@popsuper1982:/home/openstack/secret# cat disk-secret.xml
<secret ephemeral='no' private='no'>
  <usage type='volume'>
    <volume>/home/openstack/secret/encrypt.qcow2</volume>
  </usage>
</secret>
```

```
root@popsuper1982:/home/openstack/secret# virsh secret-list
 UUID                                  Usage
--------------------------------------------------------------------------------
 027f6cca-e293-45aa-a4d1-6ddbd8b7f706  volume /home/openstack/secret/encrypt.qcow2
```

```
root@popsuper1982:/home/openstack/secret# cat secret-disk-pool.xml
<pool type='dir'>
  <name>SecretDiskPool</name>
  <source>
  </source>
  <target>
    <path>/home/openstack/secret</path>
  </target>
</pool>
```

```
root@popsuper1982:/home/openstack/secret# cat secret-disk.xml
<volume>
  <name>encrypt.qcow2</name>
  <capacity>5368709120</capacity>
  <target>
    <format type='qcow2'/>
    <encryption format='qcow'>
      <secret type='passphrase' uuid='027f6cca-e293-45aa-a4d1-6ddbd8b7f706'/>
    </encryption>
  </target>
</volume>
```

3.9 Libvirt: Storage

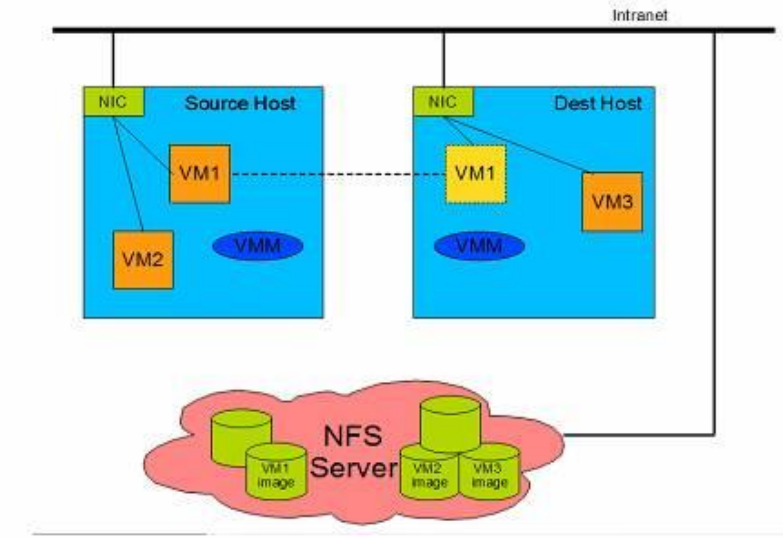
- Disk Encryption
  - 以这个disk为基础，从iso创建一虚拟机，在虚拟机xml里面也要引用secret
    - virsh edit ubuntutest4
    - virsh start ubuntutest4
  - 虚拟机安装完毕后，生成了一个加密的硬盘
  - 如果直接用qemu从硬盘启动
    - qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda encrypt.qcow2 -vnc :19 -net nic,model=virtio -net tap,ifname=tap0,script=no,downscript=no -monitor stdio
    - 是需要输入密码123456

```
<os>
  <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
  <boot dev='cdrom'/>
  <boot dev='hd'/>
</os>
```

```
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2'/>
  <source file='/home/openstack/secret/encrypt.qcow2'/>
  <target dev='vda' bus='virtio'/>
  <encryption format='qcow'>
    <secret type='passphrase' uuid='027f6cca-e293-45aa-a4d1-6ddbd8b7f706'/>
  </encryption>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</disk>
<disk type='file' device='cdrom'>
  <driver name='qemu' type='raw'/>
  <source file='/home/openstack/images/ubuntu-14.04-server-amd64.iso'/>
  <target dev='hdc' bus='ide'/>
  <readonly/>
  <address type='drive' controller='0' bus='1' target='0' unit='0'/>
</disk>
```

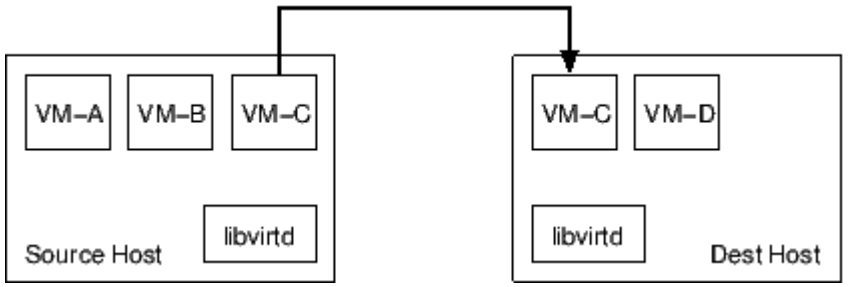
3.10 Libvirt: Migration

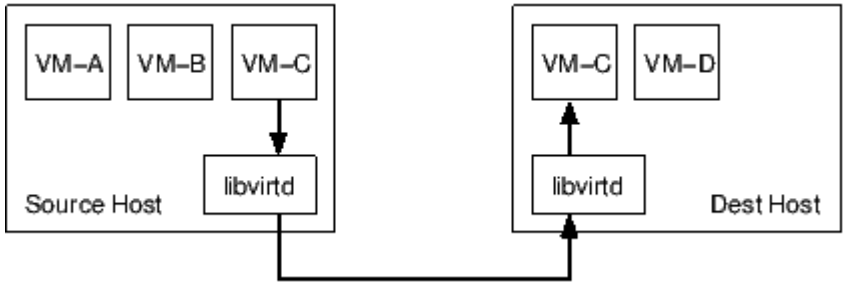
- 动态迁移Live Migration: 在保证虚拟机上服务正常运行的同时，将一个虚拟机系统从一个物理主机移动到另一个物理主机的过程。
- 动态迁移的分类
  - 共享存储 vs. 非共享存储
    - 共享存储是使用NFS，SAN等方式，使得迁移的source和destination都能访问到相同的磁盘文件
    - 非共享存储: --copy-storage-all拷贝所有的磁盘文件，--copy-storage-inc有相同的base image，则只拷贝修改的部分



3.10 Libvirt: Migration

- 动态迁移的分类
  - Native迁移 vs. tunnelled迁移
    - Native迁移：利用KVM自己的功能进行迁移，正如前面使用monitor进行迁移
    - Tunnelled迁移：通过Libvirt实现迁移，数据通过libvirt传输

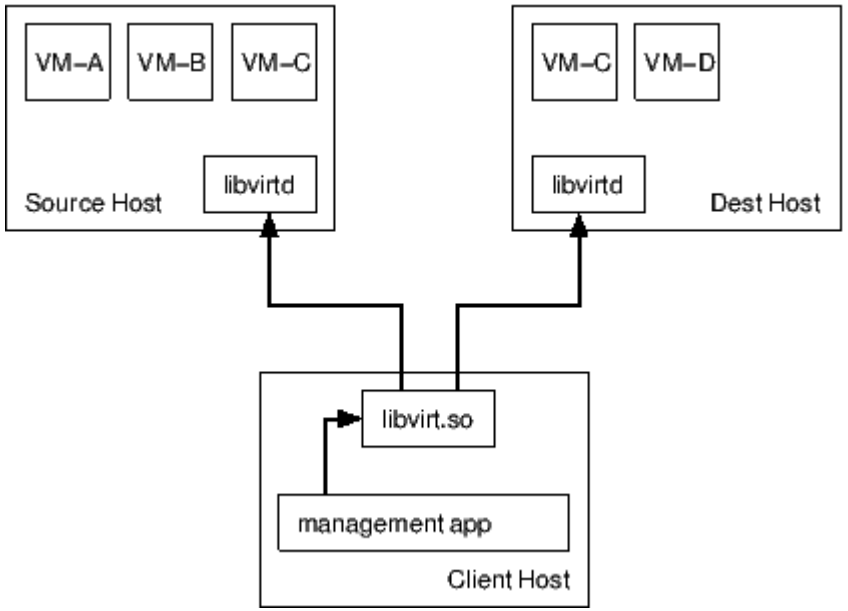




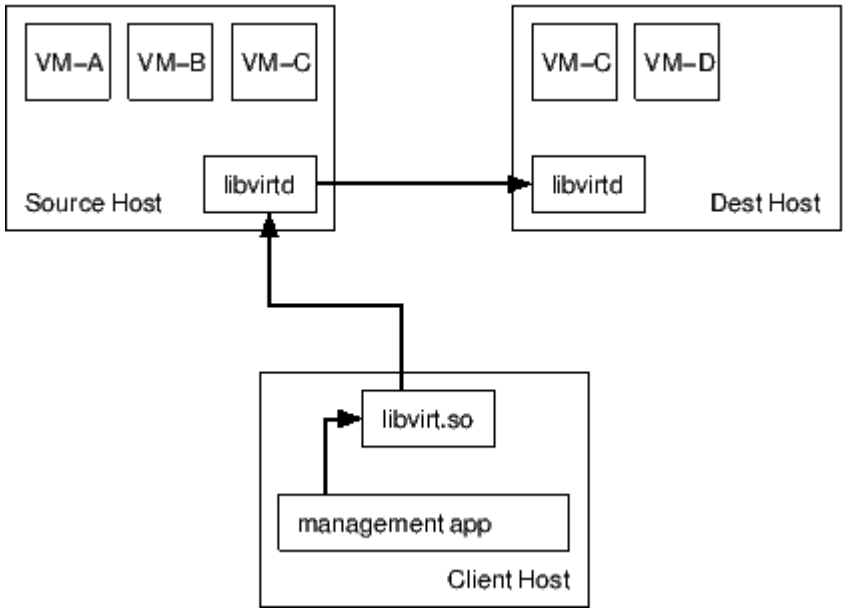
--tunnelled

3.10 Libvirt: Migration

- 动态迁移的分类
  - Direct迁移 vs. Peer to Peer迁移
    - Direct迁移：由client控制libvirtd之间的迁移，client需要同时和source和destination交互
    - Peer to Peer迁移：client仅仅和source libvirtd交互，由source负责整个迁移的过程



--direct



--p2p

3.10 Libvirt: Migration

- 首先需要使得source和destination之间的libvirt是相互通的
- 在source机器上，16.158.166.197(desthost)
  - 设置好所有的key和certificate
  - 配置/etc/libvirt/libvirtd.conf， 为了方便测试，对tcp, tls不进行密码设置
  - 配置/etc/hosts保证hostname和IP对应
  - 测试下面的命令都能通过
    - virsh -c qemu+ssh://openstack@popsuper1982/system list --all
    - virsh -c qemu+tcp://popsuper1982/system list --all
    - virsh -c qemu+tls://popsuper1982/system list --all

```
listen_tls = 1
listen_tcp = 1
tls_port = "16514"
tcp_port = "16509"
unix_sock_group = "libvirtd"
unix_sock_ro_perms = "0777"
unix_sock_rw_perms = "0770"
auth_unix_ro = "none"
auth_unix_rw = "none"
auth_tcp = "none"
auth_tls = "none"
```

```
# tree --charset ASCII /etc/pki/
/etc/pki/
|-- CA
|   `-- cacert.pem
|-- libvirt
|   |-- clientcert.pem
|   |-- private
|   |   |-- clientkey.pem
|   |   `-- serverkey.pem
|   `-- servercert.pem
`-- nssdb -> /var/lib/nssdb
```

3.10 Libvirt: Migration

- 在destination机器上，16.158.166.150(popsuper1982)
  - 设置好所有的key和certificate
  - 配置/etc/libvirt/libvirtd.conf， 为了方便测试，对tcp, tls不进行密码设置
  - 配置/etc/hosts保证hostname和IP对应
  - 测试下面的命令都能通过
    - virsh -c qemu+ssh://openstack@desthost/system list --all
    - virsh -c qemu+tcp://desthost/system list --all
    - virsh -c qemu+tls://desthost/system list –all
- 搭建一个NFS服务器，mount到source和destination机器上
  - mount 16.158.166.150:/home/openstack/nfs /mnt/migrate

3.10 Libvirt: Migration

- 在source上创建一个虚拟机ubuntutest4

```
<domain type='kvm' id='46'>
  <name>ubuntutest4</name>
  <uuid>0f0806ab-531d-6134-5def-c5b4955292ad</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
```

```
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2' cache='none'/>
      <source file='/mnt/migrate/ubuntutest4.qcow2'/>
      <target dev='vda' bus='virtio'/>
      <alias name='virtio-disk0'/>
    </disk>
    <controller type='ide' index='0'>
      <alias name='ide0'/>
    </controller>
    <controller type='usb' index='0'>
      <alias name='usb0'/>
    </controller>
    <controller type='pci' index='0' model='pci-root'>
      <alias name='pci.0'/>
    </controller>
    <interface type='network'>
      <mac address='52:54:00:e2:59:23'/>
      <source network='default'/>
      <target dev='vnet1'/>
      <model type='rtl8139'/>
      <alias name='net0'/>
    </interface>
```

```
    <serial type='pty'>
      <source path='/dev/pts/23'/>
      <target port='0'/>
      <alias name='serial0'/>
    </serial>
    <console type='pty' tty='/dev/pts/23'>
      <source path='/dev/pts/23'/>
      <target type='serial' port='0'/>
      <alias name='serial0'/>
    </console>
    <input type='mouse' bus='ps2'/>
    <input type='keyboard' bus='ps2'/>
    <graphics type='vnc' port='5907' autoport='yes' listen='0.0.0.0'>
      <listen type='address' address='0.0.0.0'/>
    </graphics>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
      <alias name='video0'/>
    </video>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
    </memballoon>
  </devices>
  <seclabel type='none' model='apparmor'/>
</domain>
```

3.10 Libvirt: Migration

- 开始迁移
  - virsh migrate --verbose --live --persistent ubuntutest4 qemu+ssh://openstack@16.158.166.197/system
- 几个注意事项：
  - 硬盘需要共享，如果ubuntutest4.qcow2有backing_file，也必须共享
  - Cache='none‘
  - Network的配置两边需要一致
  - CPU必须兼容
  - 不要有本地才有的资源，比如CD-ROM指向iso
  - VNC监听0.0.0.0

2.11 Libvirt: Hooks

- Libvirt提供hook功能，可以再下面的事件发生的时候，调用脚本做一些事情
  - 事件一：The libvirt daemon starts, stops, or reloads its configuration
  - 事件二：A QEMU guest is started or stopped
  - 事件三：A network is started or stopped or an interface is plugged/unplugged to/from the network
- Hook在目录/etc/libvirt/hooks下：
  - /etc/libvirt/hooks/daemon在事件一发生的时候被调用
  - /etc/libvirt/hooks/qemu对应事件二
  - /etc/libvirt/hooks/network对应事件三
- 脚本可以是bash也可以是python
  - #!/bin/bash
  - #!/usr/bin/python
- 脚本参数
  - Object：例如guest的名称
  - Operation：例如start
  - sub-operation
  - extra argument
  - 另外domain的xml作为stdin
- 注意：不可以在hook脚本中调用virsh的api，容易引起死锁

2.11 Libvirt: Hooks

- 简单的hook脚本，打印参数和stdin
  - /etc/libvirt/hooks/daemon
  - /etc/libvirt/hooks/qemu
  - /etc/libvirt/hooks/network
- 要加载hook脚本，必须stop然后start libvirt-bin
  - service libvirt-bin stop
  - service libvirt-bin start
  - 不可以restart
- 测试脚本
  - tail -f /tmp/hook.log

```
root@popsuper1982:/etc/libvirt/hooks# ls
daemon  network  qemu
root@popsuper1982:/etc/libvirt/hooks# cat daemon
#!/bin/bash
echo $0 $@ >>/tmp/hook.log
root@popsuper1982:/etc/libvirt/hooks# cat network
#!/bin/bash
echo $0 $@ >>/tmp/hook.log
root@popsuper1982:/etc/libvirt/hooks# cat qemu
#!/bin/bash
echo $0 $@ >>/tmp/hook.log
xml=$(cat /dev/stdin)
echo $xml >> /tmp/hook.log
```

```
root@popsuper1982:/etc/libvirt/hooks# service libvirt-bin stop
libvirt-bin stop/waiting
root@popsuper1982:/etc/libvirt/hooks# service libvirt-bin start
libvirt-bin start/running, process 315
```

```
/etc/libvirt/hooks/daemon - shutdown - shutdown
/etc/libvirt/hooks/daemon - start - start
/etc/libvirt/hooks/network default plugged begin -
/etc/libvirt/hooks/qemu ubuntutest4 reconnect begin -
<domain type='kvm' id='11'> <name>ubuntutest4</name> <uuid>0f0806ab-531d-6134-5def-c5b4955292ad</uuid> <memory unit='KiB'>1048576</memory> <currentMemory unit='KiB'>1048576
</currentMemory> <vcpu placement='static'>1</vcpu> <resource> <partition>/machine</partition> </resource> <os> <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type> <bo
ot dev='hd'/> </os> <features> <acpi/> <apic/> <pae/> </features> <clock offset='utc'/> <on_poweroff>destroy</on_poweroff> <on_reboot>restart</on_reboot> <on_crash>restart<
/on_crash> <devices> <emulator>/usr/bin/qemu-system-x86_64</emulator> <disk type='file' device='disk'> <driver name='qemu' type='qcow2'/> <source file='/home/openstack/imag
es/ubuntutest4.qcow2'/> <target dev='vda' bus='virtio'/> <alias name='virtio-disk0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/> </disk> <c
ontroller type='pci' index='0' model='pci-root'> <alias name='pci.0'/> </controller> <controller type='usb' index='0'> <alias name='usb0'/> <address type='pci' domain='0x00
00' bus='0x00' slot='0x01' function='0x2'/> </controller> <interface type='network'> <mac address='52:54:00:e2:59:23'/> <source network='default'/> <target dev='vnet0'/> <m
odel type='rtl8139'/> <alias name='net0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/> </interface> <serial type='pty'> <source path='/dev/p
ts/3'/> <target port='0'/> <alias name='serial0'/> </serial> <console type='pty' tty='/dev/pts/3'> <source path='/dev/pts/3'/> <target type='serial' port='0'/> <alias name=
'serial0'/> </console> <input type='mouse' bus='ps2'/> <input type='keyboard' bus='ps2'/> <graphics type='vnc' port='5900' autoport='yes' listen='0.0.0.0'> <listen type='ad
dress' address='0.0.0.0'/> </graphics> <video> <model type='cirrus' vram='9216' heads='1'/> <alias name='video0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x02
' function='0x0'/> </video> <memballoon model='virtio'> <alias name='balloon0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/> </memballoon> <
/devices> <seclabel type='none'/> </domain>
```

2.11 Libvirt: Hooks

```
root@popsuper1982:/etc/libvirt/hooks# virsh destroy ubuntutest4
Domain ubuntutest4 destroyed
```

```
/etc/libvirt/hooks/qemu ubuntutest4 stopped end -
<domain type='kvm'> <name>ubuntutest4</name> <uuid>0f0806ab-531d-6134-5def-c5b4955292ad</uuid> <memory unit='KiB'>1048576</memory> <currentMemory unit='KiB'>1048576</curren
tMemory> <vcpu placement='static'>1</vcpu> <resource> <partition>/machine</partition> </resource> <os> <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type> <boot dev='
hd'/> </os> <features> <acpi/> <apic/> <pae/> </features> <clock offset='utc'/> <on_poweroff>destroy</on_poweroff> <on_reboot>restart</on_reboot> <on_crash>restart</on_cras
h> <devices> <emulator>/usr/bin/qemu-system-x86_64</emulator> <disk type='file' device='disk'> <driver name='qemu' type='qcow2'/> <source file='/home/openstack/images/ubunt
utest4.qcow2'/> <target dev='vda' bus='virtio'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/> </disk> <controller type='pci' index='0' model=
'pci-root'/> <controller type='usb' index='0'> <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'/> </controller> <interface type='network'> <mac add
ress='52:54:00:e2:59:23'/> <source network='default'/> <model type='rtl8139'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/> </interface> <ser
ial type='pty'> <target port='0'/> </serial> <console type='pty'> <target type='serial' port='0'/> </console> <input type='mouse' bus='ps2'/> <input type='keyboard' bus='ps
2'/> <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'> <listen type='address' address='0.0.0.0'/> </graphics> <video> <model type='cirrus' vram='9216' heads='
1'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/> </video> <memballoon model='virtio'> <address type='pci' domain='0x0000' bus='0x00' slot='0
x04' function='0x0'/> </memballoon> </devices> <seclabel type='none'/> </domain>
/etc/libvirt/hooks/network default unplugged begin -
/etc/libvirt/hooks/qemu ubuntutest4 release end -
<domain type='kvm'> <name>ubuntutest4</name> <uuid>0f0806ab-531d-6134-5def-c5b4955292ad</uuid> <memory unit='KiB'>1048576</memory> <currentMemory unit='KiB'>1048576</curren
tMemory> <vcpu placement='static'>1</vcpu> <resource> <partition>/machine</partition> </resource> <os> <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type> <boot dev='
hd'/> </os> <features> <acpi/> <apic/> <pae/> </features> <clock offset='utc'/> <on_poweroff>destroy</on_poweroff> <on_reboot>restart</on_reboot> <on_crash>restart</on_cras
h> <devices> <emulator>/usr/bin/qemu-system-x86_64</emulator> <disk type='file' device='disk'> <driver name='qemu' type='qcow2'/> <source file='/home/openstack/images/ubunt
utest4.qcow2'/> <target dev='vda' bus='virtio'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/> </disk> <controller type='pci' index='0' model=
'pci-root'/> <controller type='usb' index='0'> <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'/> </controller> <interface type='network'> <mac add
ress='52:54:00:e2:59:23'/> <source network='default'/> <model type='rtl8139'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/> </interface> <ser
ial type='pty'> <target port='0'/> </serial> <console type='pty'> <target type='serial' port='0'/> </console> <input type='mouse' bus='ps2'/> <input type='keyboard' bus='ps
2'/> <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'> <listen type='address' address='0.0.0.0'/> </graphics> <video> <model type='cirrus' vram='9216' heads='
1'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/> </video> <memballoon model='virtio'> <address type='pci' domain='0x0000' bus='0x00' slot='0
x04' function='0x0'/> </memballoon> </devices> <seclabel type='none'/> </domain>
```

2.11 Libvirt: Hooks

```
root@popsuper1982:/etc/libvirt/hooks# virsh start ubuntutest4
Domain ubuntutest4 started
```

```
/etc/libvirt/hooks/qemu ubuntutest4 prepare begin -
<domain type='kvm' id='13'> <name>ubuntutest4</name> <uuid>0f0806ab-531d-6134-5def-c5b4955292ad</uuid> <memory unit='KiB'>1048576</memory> <currentMemory unit='KiB'>1048576
</currentMemory> <vcpu placement='static'>1</vcpu> <resource> <partition>/machine</partition> </resource> <os> <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type> <bo
ot dev='hd'/> </os> <features> <acpi/> <apic/> <pae/> </features> <clock offset='utc'/> <on_poweroff>destroy</on_poweroff> <on_reboot>restart</on_reboot> <on_crash>restart<
/on_crash> <devices> <emulator>/usr/bin/qemu-system-x86_64</emulator> <disk type='file' device='disk'> <driver name='qemu' type='qcow2'/> <source file='/home/openstack/imag
es/ubuntutest4.qcow2'/> <target dev='vda' bus='virtio'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/> </disk> <controller type='pci' index='0
' model='pci-root'/> <controller type='usb' index='0'> <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'/> </controller> <interface type='network'>
<mac address='52:54:00:e2:59:23'/> <source network='default'/> <model type='rtl8139'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/> </interfa
ce> <serial type='pty'> <target port='0'/> </serial> <console type='pty'> <target type='serial' port='0'/> </console> <input type='mouse' bus='ps2'/> <input type='keyboard'
 bus='ps2'/> <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'> <listen type='address' address='0.0.0.0'/> </graphics> <video> <model type='cirrus' vram='9216'
 heads='1'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/> </video> <memballoon model='virtio'> <address type='pci' domain='0x0000' bus='0x00'
 slot='0x04' function='0x0'/> </memballoon> </devices> <seclabel type='none'/> </domain>
/etc/libvirt/hooks/network default plugged begin -
/etc/libvirt/hooks/qemu ubuntutest4 start begin -
<domain type='kvm' id='13'> <name>ubuntutest4</name> <uuid>0f0806ab-531d-6134-5def-c5b4955292ad</uuid> <memory unit='KiB'>1048576</memory> <currentMemory unit='KiB'>1048576
</currentMemory> <vcpu placement='static'>1</vcpu> <resource> <partition>/machine</partition> </resource> <os> <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type> <bo
ot dev='hd'/> </os> <features> <acpi/> <apic/> <pae/> </features> <clock offset='utc'/> <on_poweroff>destroy</on_poweroff> <on_reboot>restart</on_reboot> <on_crash>restart<
/on_crash> <devices> <emulator>/usr/bin/qemu-system-x86_64</emulator> <disk type='file' device='disk'> <driver name='qemu' type='qcow2'/> <source file='/home/openstack/imag
es/ubuntutest4.qcow2'/> <target dev='vda' bus='virtio'/> <alias name='virtio-disk0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/> </disk> <c
ontroller type='pci' index='0' model='pci-root'> <alias name='pci.0'/> </controller> <controller type='usb' index='0'> <alias name='usb0'/> <address type='pci' domain='0x00
00' bus='0x00' slot='0x01' function='0x2'/> </controller> <interface type='network'> <mac address='52:54:00:e2:59:23'/> <source network='default'/> <target dev='vnet0'/> <m
odel type='rtl8139'/> <alias name='net0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/> </interface> <serial type='pty'> <target port='0'/> <
alias name='serial0'/> </serial> <console type='pty'> <target type='serial' port='0'/> <alias name='serial0'/> </console> <input type='mouse' bus='ps2'/> <input type='keybo
ard' bus='ps2'/> <graphics type='vnc' port='5900' autoport='yes' listen='0.0.0.0'> <listen type='address' address='0.0.0.0'/> </graphics> <video> <model type='cirrus' vram=
'9216' heads='1'/> <alias name='video0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/> </video> <memballoon model='virtio'> <alias name='ball
oon0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/> </memballoon> </devices> <seclabel type='none'/> </domain>
/etc/libvirt/hooks/qemu ubuntutest4 started begin -
<domain type='kvm' id='13'> <name>ubuntutest4</name> <uuid>0f0806ab-531d-6134-5def-c5b4955292ad</uuid> <memory unit='KiB'>1048576</memory> <currentMemory unit='KiB'>1048576
</currentMemory> <vcpu placement='static'>1</vcpu> <resource> <partition>/machine</partition> </resource> <os> <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type> <bo
ot dev='hd'/> </os> <features> <acpi/> <apic/> <pae/> </features> <clock offset='utc'/> <on_poweroff>destroy</on_poweroff> <on_reboot>restart</on_reboot> <on_crash>restart<
/on_crash> <devices> <emulator>/usr/bin/qemu-system-x86_64</emulator> <disk type='file' device='disk'> <driver name='qemu' type='qcow2'/> <source file='/home/openstack/imag
es/ubuntutest4.qcow2'/> <target dev='vda' bus='virtio'/> <alias name='virtio-disk0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/> </disk> <c
ontroller type='pci' index='0' model='pci-root'> <alias name='pci.0'/> </controller> <controller type='usb' index='0'> <alias name='usb0'/> <address type='pci' domain='0x00
00' bus='0x00' slot='0x01' function='0x2'/> </controller> <interface type='network'> <mac address='52:54:00:e2:59:23'/> <source network='default'/> <target dev='vnet0'/> <m
odel type='rtl8139'/> <alias name='net0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/> </interface> <serial type='pty'> <source path='/dev/p
ts/3'/> <target port='0'/> <alias name='serial0'/> </serial> <console type='pty' tty='/dev/pts/3'> <source path='/dev/pts/3'/> <target type='serial' port='0'/> <alias name=
'serial0'/> </console> <input type='mouse' bus='ps2'/> <input type='keyboard' bus='ps2'/> <graphics type='vnc' port='5900' autoport='yes' listen='0.0.0.0'> <listen type='ad
dress' address='0.0.0.0'/> </graphics> <video> <model type='cirrus' vram='9216' heads='1'/> <alias name='video0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x02
' function='0x0'/> </video> <memballoon model='virtio'> <alias name='balloon0'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/> </memballoon> <
/devices> <seclabel type='none'/> </domain>
```

2.11 Libvirt: Hooks

```
root@popsuper1982:/etc/libvirt/hooks# cat qemu
#!/bin/bash
DOMAIN=$1; shift
xml=$(cat /dev/stdin)
case "$*" in
    "started begin -" )
        mkdir -p /etc/libvirt/hooks/qemu.d/$DOMAIN
        echo $xml > /etc/libvirt/hooks/qemu.d/$DOMAIN/${DOMAIN}.xml
        python /etc/libvirt/hooks/qemu.d/disable_icmp.py $DOMAIN >> /tmp/hook.log 2>&1
        python /etc/libvirt/hooks/qemu.d/update_cpu_share.py $DOMAIN >> /tmp/hook.log 2>&1
        python /etc/libvirt/hooks/qemu.d/add_tc.py $DOMAIN >> /tmp/hook.log 2>&1
    ;;
    "stopped end -" )
        python /etc/libvirt/hooks/qemu.d/enable_icmp.py $DOMAIN >> /tmp/hook.log 2>&1
        rm -rf /etc/libvirt/hooks/qemu.d/$DOMAIN
    ;;
esac
```

- 复杂的hook脚本
  - 禁止ICMP
  - 设置CPU Share
  - 设置网络带宽
- 修改/etc/libvirt/hooks/qemu
  - 处理started和stopped事件
  - 在started事件发生后，也即虚拟机创建后
    - 创建一个文件夹，将domain的xml放在文件夹里面
    - 然后调用python脚本
      - /etc/libvirt/hooks/qemu.d/disable_icmp.py
      - /etc/libvirt/hooks/qemu.d/update_cpu_share.py
      - /etc/libvirt/hooks/qemu.d/add_tc.py
  - 在stopped事件发生后，也即虚拟机关闭后
    - 调用脚本/etc/libvirt/hooks/qemu.d/enable_icmp.py
    - 删除domain文件夹

2.11 Libvirt: Hooks

- /etc/libvirt/hooks/qemu.d/disable_icmp.py
  - 解析domain的xml文件，得到domain的名称和网卡名称
  - 调用iptables命令

```
root@popsuper1982:/etc/libvirt/hooks# cat /etc/libvirt/hooks/qemu.d/disable_icmp.py
#!/usr/bin/python
import sys
import os
from xml.etree import ElementTree

domain = sys.argv[1]
root = ElementTree.parse(r"/etc/libvirt/hooks/qemu.d/" + domain + "/" + domain + ".xml")

domain_name = root.find('name').text
domain_interface = root.find('devices/interface/target').attrib['dev']

print "enable icmp of " + domain_name + " on device " + domain_interface

os.system("iptables -A INPUT -i " + domain_interface + " -p icmp -m icmp --icmp-type 8 -j DROP")
```

2.11 Libvirt: Hooks

- /etc/libvirt/hooks/qemu.d/enable_icmp.py
  - 解析domain的xml文件，得到domain的名称和网卡名称
  - 调用iptables命令

```
root@popsuper1982:/etc/libvirt/hooks# cat /etc/libvirt/hooks/qemu.d/enable_icmp.py
#!/usr/bin/python
import sys
import os
from xml.etree import ElementTree

domain = sys.argv[1]
root = ElementTree.parse(r"/etc/libvirt/hooks/qemu.d/" + domain + "/" + domain + ".xml")

domain_name = root.find('name').text
domain_interface = root.find('devices/interface/target').attrib['dev']

print "enable icmp of " + domain_name + " on device " + domain_interface

os.system("iptables -D INPUT -i " + domain_interface + " -p icmp -m icmp --icmp-type 8 -j DROP")
```

2.11 Libvirt: Hooks

- /etc/libvirt/hooks/qemu.d/update_cpu_share.py
  - 解析domain的xml文件，得到domain的名称和网卡名称
  - 将2048写入cpu.shares文件，默认为1024

```
root@popsuper1982:/etc/libvirt/hooks# cat /etc/libvirt/hooks/qemu.d/update_cpu_share.py
#!/usr/bin/python
import sys
import os
from xml.etree import ElementTree

domain = sys.argv[1]
root = ElementTree.parse(r"/etc/libvirt/hooks/qemu.d/" + domain + "/" + domain + ".xml")

domain_name = root.find('name').text

cgroup_file = "/sys/fs/cgroup/cpu/machine/" + domain_name + ".libvirt-qemu/cpu.shares"

print "update cgroup " + cgroup_file

os.system("echo 2048 > " + cgroup_file)
```

2.11 Libvirt: Hooks

- /etc/libvirt/hooks/qemu.d/add_tc.py
  - 解析domain的xml文件，得到domain的名称和网卡名称
  - 调用tc添加qdisc到虚拟网卡

```
root@popsuper1982:/etc/libvirt/hooks# cat /etc/libvirt/hooks/qemu.d/add_tc.py
#!/usr/bin/python

import sys
import os
from xml.etree import ElementTree

domain = sys.argv[1]
root = ElementTree.parse(r"/etc/libvirt/hooks/qemu.d/" + domain + "/" + domain + ".xml")

domain_name = root.find('name').text
domain_interface = root.find('devices/interface/target').attrib['dev']

print "add tc of " + domain_name + " on device " + domain_interface

os.system("tc qdisc add dev " + domain_interface + " root handle 1: htb")
os.system("tc class add dev " + domain_interface + " parent 1: classid 1:1 htb rate 100kbps ceil 100kbps")
```

2.11 Libvirt: Hooks

- 测试：
  - 启动虚拟机virsh start ubuntutest4
    - 在tail -f /tmp/hook.log中

```
/etc/libvirt/hooks/network default plugged begin -
enable icmp of ubuntutest4 on device vnet0
update cgroup /sys/fs/cgroup/cpu/machine/ubuntutest4.libvirt-qemu/cpu.shares
add tc of ubuntutest4 on device vnet0
```

    - 运行iptables –nvL

```
root@popsuper1982:/etc/libvirt/hooks# iptables -nvL
Chain INPUT (policy ACCEPT 738 packets, 90901 bytes)
 pkts bytes target     prot opt in     out     source               destination
   16   960 ACCEPT     udp  --  virbr0 *       0.0.0.0/0            0.0.0.0/0           udp dpt:53
    0     0 ACCEPT     tcp  --  virbr0 *       0.0.0.0/0            0.0.0.0/0           tcp dpt:53
    8  2624 ACCEPT     udp  --  virbr0 *       0.0.0.0/0            0.0.0.0/0           udp dpt:67
    0     0 ACCEPT     tcp  --  virbr0 *       0.0.0.0/0            0.0.0.0/0           tcp dpt:67
 684K 1442M libvirt-host-in  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DROP       icmp --  vnet0  *       0.0.0.0/0            0.0.0.0/0           icmptype 8
```

    - 查看/sys/fs/cgroup/cpu/machine/ubuntutest4.libvirt-qemu/cpu.shares
    - 查看tc qdisc show dev vnet0

```
root@popsuper1982:/etc/libvirt/hooks# cat /sys/fs/cgroup/cpu/machine/ubuntutest4.libvirt-qemu/cpu.shares
2048
root@popsuper1982:/etc/libvirt/hooks# tc qdisc show dev vnet0
qdisc htb 1: root refcnt 2 r2q 10 default 0 direct_packets_stat 203 direct_qlen 500
```

    - 文件夹也创建了

```
root@popsuper1982:/etc/libvirt/hooks# tree --charset ASCII qemu.d/
qemu.d/
|-- add_tc.py
|-- disable_icmp.py
|-- enable_icmp.py
|-- ubuntutest4
|   `-- ubuntutest4.xml
`-- update_cpu_share.py
```

2.11 Libvirt: Hooks

- 测试：
  - 删除虚拟机virsh destroy ubuntutest4
    - 在tail -f /tmp/hook.log中
    - 运行iptables –nvL

```
enable icmp of ubuntutest4 on device vnet0
/etc/libvirt/hooks/network default unplugged begin -
```

```
root@popsuper1982:/etc/libvirt/hooks# iptables -nvL
Chain INPUT (policy ACCEPT 394 packets, 52629 bytes)
 pkts bytes target     prot opt in     out     source               destination
   16   960 ACCEPT     udp  --  virbr0 *       0.0.0.0/0            0.0.0.0/0           udp dpt:53
    0     0 ACCEPT     tcp  --  virbr0 *       0.0.0.0/0            0.0.0.0/0           tcp dpt:53
    8  2624 ACCEPT     udp  --  virbr0 *       0.0.0.0/0            0.0.0.0/0           udp dpt:67
    0     0 ACCEPT     tcp  --  virbr0 *       0.0.0.0/0            0.0.0.0/0           tcp dpt:67
 685K 1442M libvirt-host-in  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

    - 文件夹也删除了

```
root@popsuper1982:/etc/libvirt/hooks# tree --charset ASCII qemu.d/
qemu.d/
|-- add_tc.py
|-- disable_icmp.py
|-- enable_icmp.py
`-- update_cpu_share.py
```