虚拟化技术基础：

cpu虚拟化：

模拟：emulation

虚拟：virtulization

完全虚拟化（full-virtulization）

BT: 二进制翻译 （软件）

HVM：硬件辅助的虚拟化 （硬件）

半虚拟化(para-virtulization)

Memory:

进程：线性地址空间

内核：物理地址空间

MMU Virtulization

Intel: EPT, Extended Page Table

AMD: NTP, Nested Page Table

TLB virtulization

tagged TLB

I/O:

外存：

硬盘、光盘、U盘

网络设备：

网卡

显示设备：

VGA: frame buffer机制

键盘鼠标：

ps/2, usb

I/O虚拟化的方式：

模拟: 完全使用软件来模拟真实硬件

半虚拟化化:

IO-through: IO透传

Intel: VT-d

基于北桥的硬件辅助的虚拟化技术；

两种实现方式：

Type-I:

xen, vmware ESX/ESXi

Type-II:

kvm, vmware workstation, virtualbox

Intel硬件辅助的虚拟化：

CPU: vt-x, EPT, tagged-TLB

IO/CPU: vt-d, IOV, VMDq

第一类：跟处理器相关：vt-x

第二类：跟芯片相关：vt-d

第三类：跟IO相关：VMDq和SR-IOV

QEMU, virtio

虚拟化技术的分类：

模拟：著名的模拟器，PearPC, Bochs, QEMU

完全虚拟化：也称为native virtulization

两种加速方式：

BT

HVM

VMware Workstation, VMware Server, Parallels Desktop, KVM, Xen(HVM)

半虚拟化：para-virtualization

xen, uml(user-mode linux)

OS级别的虚拟化:

OpenVZ, lxc

Solaris Containers

FreeBSD jails

库虚拟化：

wine

应用程序虚拟化：

jvm

虚拟化网络：

nat mode

bridge mode

routed mode

isolation mode

使用brctl的配置的过程示例：

# brctl addbr br0

# brctl stp br0 on

# ifconfig eth0 0 up

# brctl addif br0 eth0

# ifconfig br0 IP/NETMASK up

# route add default gw GW

补充资料：TUN与TAP

在计算机网络中，TUN与TAP是操作系统内核中的虚拟网络设备。不同于普通靠硬件网路板卡实现的设备，这些虚拟的网络设备全部用软件实现，并向运行于操作系统上的软件提供与硬件的网络设备完全相同的功能。

TAP等同于一个以太网设备，它操作第二层数据包如以太网数据帧。TUN模拟了网络层设备，操作第三层数据包比如IP数据封包。

操作系统通过TUN/TAP设备向绑定该设备的用户空间的程序发送数据，反之，用户空间的程序也可以像操作硬件网络设备那样，通过TUN/TAP设备发送数据。在后种情况下，TUN/TAP设备向操作系统的网络栈投递（或“注入”）数据包，从而模拟从外部接受数据的过程。

回顾：

虚拟化技术的分类：

(1) 模拟：Emulation；

Qemu, PearPC, Bochs

(2) 完全虚拟化：Full Virtualization, Native Virtualization

HVM

VMware Workstation, VirtualBox, VMWare Server, Parallels Desktop, KVM, XEN

(3) 半虚拟化：ParaVirtualization

GuestOS: 知晓自己是运行Virtualization

Hypercall

解决安装：Xen, UML(User-Mode Linux)

(4) OS级别的虚拟化：

将用户空间分隔为多个，彼此间互相隔离；

容器级虚拟化

OpenVZ, LXC(LinuX Container), libcontainer, Virtuozzo, Linux V Servers

(5) 库级别虚拟化

WINE

Type-I, Type-II

IaaS: Infrastructure

PaaS: Platfrom

Xen：

剑桥大学，开源VMM；

Xen组成部分：

(1) Xen Hypervisor

分配CPU、Memory、Interrupt

(2) Dom0

特权域，I/O分配

网络设备

net-front(GuestOS), net-backend

块设备

block-front(GuestOS), block-backend

Linux Kernel:

2.6.37：开始支持运行Dom0

3.0：对关键特性进行了优化

提供管理DomU工具栈

用于实现对虚拟进行添加、启动、快照、停止、删除等操作；

(3) DomU

非特权域，根据其虚拟化方式实现，有多种类型

PV

HVM

PV on HVM

Xen的PV技术：

不依赖于CPU的HVM特性，但要求GuestOS的内核作出修改以知晓自己运行于PV环境；

运行于DomU中的OS：Linux(2.6.24+), NetBSD, FreeBSD, OpenSolaris

Xen的HVM技术：

依赖于Intel VT或AMD AMD-V，还要依赖于Qemu来模拟IO设备；

运行于DomU中的OS：几乎所有支持此X86平台的；

PV on HVM：

CPU为HVM模式运行

IO设备为PV模式运行

运行于DomU中的OS: 只要OS能驱动PV接口类型的IO设备；

net-front, blk-front

Xen的工具栈：

xm/xend：在Xen Hypervisor的Dom0中要启动xend服务

xm：命令行管理工具，有诸多子命令；

create, destroy, stop, pause...

xl: 基于libxenlight提供的轻量级的命令行工具栈；

xe/xapi：提供了对xen管理的api，因此多用于cloud环境；Xen Server, XCP

virsh/libvirt：

XenStore：

为各Domain提供的共享信息存储空间；有着层级结构的名称空间；位于Dom0

CentOS对Xen的支持：

RHEL 5.7-：默认的虚拟化技术为xen；

kernel version: 2.6.18

kernel-

kernel-xen

RHEL 6+：仅支持kvm

Dom0: 不支持

DomU: 支持

如何在CentOS 6.6上使用Xen：

(1) 编译3.0以上版本的内核，启动对Dom0的支持；

(2) 编译xen程序；

制作好相关程序包的项目：

xen4centos

xen made easy

演示：使用Xen4CentOS

安装完成后对grub的配置示例：

(1) title CentOS (3.18.12-11.el6.x86\_64)

　　　root (hd0,0)

　　　kernel /xen.gz dom0\_mem=1024M cpufreq=xen dom0\_max\_vcpus=2 dom0\_vcpus\_pin

　　　module /vmlinuz-3.18.12-11.el6.x86\_64 ro root=/dev/mapper/vg0-root rd\_NO\_LUKS rd\_NO\_DM LANG=en\_US.UTF-8 rd\_LVM\_LV=vg0/swap rd\_NO\_MD SYSFONT=latarcyrheb-sun16 crashkernel=auto rd\_LVM\_LV=vg0/root KEYBOARDTYPE=pc KEYTABLE=us rhgb crashkernel=auto quiet rhgb quiet

module /initramfs-3.18.12-11.el6.x86\_64.img

(2) title CentOS (3.4.46-8.el6.centos.alt.x86\_64)

　　　root (hd0,0)

　　　kernel /xen.gz dom0\_mem=1024M,max:1024M loglvl=all guest\_loglvl=all

　　　module /vmlinuz-3.4.46-8.el6.centos.alt.x86\_64 ro root=/dev/mapper/vg\_xen01-lv\_root rd\_LVM\_LV=vg\_xen01/lv\_swap rd\_NO\_LUKS KEYBOARDTYPE=pc KEYTABLE=uk rd\_NO\_MD LANG=en\_GB rd\_LVM\_LV=vg\_xen01/lv\_root SYSFONT=latarcyrheb-sun16 crashkernel=auto rd\_NO\_DM rhgb quiet

　　　module /initramfs-3.4.46-8.el6.centos.alt.x86\_64.img

(3) title CentOS (3.7.10-1.el6xen.x86\_64)

root (hd0,0)

kernel /xen.gz dom0\_mem=1024M,max:1024M dom0\_max\_vcpus=2 dom0\_vcpus\_pin cpufreq=xen

module /vmlinuz-3.7.10-1.el6xen.x86\_64 ro root=/dev/mapper/vg0-root rd\_NO\_LUKS rd\_NO\_DM LANG=en\_US.UTF-8 rd\_LVM\_LV=vg0/swap rd\_NO\_MD SYSFONT=latarcyrheb-sun16 crashkernel=auto rd\_LVM\_LV=vg0/root KEYBOARDTYPE=pc KEYTABLE=us rhgb crashkernel=auto quiet rhgb quiet

module /initramfs-3.7.10-1.el6xen.x86\_64.img

grub.conf boot options(官方文档)：http://xenbits.xen.org/docs/unstable/misc/xen-command-line.html

官方Man手册：http://wiki.xenproject.org/wiki/Xen\_Man\_Pages

工具栈：

xm/xend

xl

xl list：显示Domain的相关信息

xen虚拟状态：

r: running

b: 阻塞

p: 暂停

s: 停止

c: 崩溃

d: dying, 正在关闭的过程中

如何创建xen pv模式：

1、kernel

2、initrd或initramfs

3、DomU内核模块

4、根文件系统

5、swap设备

将上述内容定义在DomU的配置文件

注意：xm与xl启动DomU使用的配置文件略有不同；

对于xl而言，其创建DomU使用的配置指令可通过“man xl.cfg”获取

常用指令：

name: 域惟一的名称

builder：指明虚拟机的类型，generic表示pv，hvm表示hvm

vcpus：虚拟cpu个数；

maxcpus：最大虚拟cpu个数

cpus：vcpu可运行于其上物理CPU列表

memory=MBYTES: 内存大小

maxmem=MBYTES：可以使用的最大内存空间

on\_poweroff：指明关机时采取的action

destroy, restart, preserve

on\_reboot="ACTION": 指明“重启”DomU时采取的action

on\_crash="ACTION"：虚拟机意外崩溃时采取的action

uuid：DomU的惟一标识；

disk=[ "DISK\_SPEC\_STRING", "DISK\_SPEC\_STRING", ...]: 指明磁盘设备，列表，

vif=[ "NET\_SPEC\_STRING", "NET\_SPEC\_STRING", ...]：指明网络接口，列表，

vfb=[ "VFB\_SPEC\_STRING", "VFB\_SPEC\_STRING", ...]：指明virtual frame buffer，列表；

pci=[ "PCI\_SPEC\_STRING", "PCI\_SPEC\_STRING", ... ]： pci设备的列表

PV模式专用指令：

kernel="PATHNAME"：内核文件路径，此为Dom0中的路径；

ramdisk="PATHNAME"：为kernel指定内核提供的ramdisk文件路径；

root="STRING"：指明根文件系统；

extra="STRING"：额外传递给内核引导时使用的参数；

bootloader="PROGRAM"：如果DomU使用自己的kernel及ramdisk，此时需要一个Dom0中的应用程序来实现其bootloader功能；

磁盘参数指定方式：

官方文档：http://xenbits.xen.org/docs/unstable/misc/xl-disk-configuration.txt

[<target>, [<format>, [<vdev>, [<access>]]]]

<target>表示磁盘映像文件或设备文件路径：/images/xen/linux.img，/dev/myvg/linux

<format>表示磁盘格式，如果映像文件，有多种格式，例如raw, qcow, qcow2

vdev: 此设备在DomU被识别为硬件设备类型，支持hd[x], xvd[x], sd[x]

access: 访问权限，

ro, r: 只读

rw, w: 读写

disk=[ "/images/xen/linux.img,raw,xvda,rw", ]

使用qemu-img管理磁盘映像：

create [-f fmt] [-o options] filename [size]

可创建sparse格式的磁盘映像文件

示例：

创建一个pv格式的vm：

(1) 准备磁盘映像文件

qemu-img create -f raw -o size=2G /images/xen/busybox.img

mke2fs -t ext /images/xen/busybox.img

(2) 提供根文件系统

编译busybox，并复制到busybox.img映像中

mount -o /images/xen/busybox.img /mnt

cp -a $BUSYBOX/\_install/\* /mnt

mkdir /mnt/{proc,sys,dev,var}

(3) 提供配置DomU配置文件

name = "busybox-001"

kernel = "/boot/vmlinuz"

ramdisk = "/boot/initramfs.img"

extra = "selinux=0 init=/bin/sh"

memory = 256

vcpus = 2

disk= [ '/images/xen/busybox.img,raw,xvda,rw' ]

root = "/dev/xvda ro"

(4) 启动实例：

xl [-v] create <DomU\_Config\_file> -n

xl create <DomU\_Config\_file> -c

Ctrl + ]

如何配置网络接口：

vif = [ '<vifspec>', '<vifspec>', ... ]

vifspec: [<key>=<value>|<flag>,]

常用的key：

mac=：指定mac地址，要以“00:16:3e”开头；

bridge=<bridge>：指定此网络接口在Dom0被关联至哪个桥设备上；

model=<MODEL>:

vifname=: 接口名称，在Dom0中显示的名称；

script=：执行的脚本；

ip=：指定ip地址，会注入到DomU中；

rate=： 指明设备传输速率，通常为"#UNIT/s"格式

UNIT: GB, MB, KB, B for bytes.

Gb, Mb, Kb, b for bits.

官方文档：http://xenbits.xen.org/docs/unstable/misc/xl-network-configuration.html

注意：上述实验配置网络时出问题，可考虑内核版本降级至3.7.4, Sources/6.x86\_64/xen-4.1；

网络接口启用：

vif = [ 'bridge=xenbr0', ]

xl的其它常用命令：

shutdown: 关机

reboot：重启

pause: 暂停

unpause: 解除暂停

save: 将DomU的内存中的数据转存至指定的磁盘文件中；

xl [-vf] save [options] <Domain> <CheckpointFile> [<ConfigFile>]

restore: 从指定的磁盘文件中恢复DomU内存数据；

xl [-vf] restore [options] [<ConfigFile>] <CheckpointFile>

vcpu-list

vcpu-pin

vcpu-set

info: 当前xen hypervisor的摘要信息

domid

domname

top: 查看domain资源占用排序状态的命令

network-list: 查看指定域使用网络及接口；

network-attach

network-detach

block-list: 查看指定域使用的块设备的列表；

block-attach

block-detach

uptime: 运行时长

使用DomU自有kernel来启动运行DomU：

制作磁盘映像文件：

losetup - set up and control loop devices

losetup -a: 显示所有已用的loop设备相关信息

losetup -f: 显示第一个空闲的loop设备文件

使用xl命令进行创建虚拟机并完成CentOS 6.6的系统安装

(1) 创建磁盘映像文件；

(2) 获取安装指定版本的系统所需要kernel及initrd文件；

(3) 创建DomU配置文件，示例如下：

name = "centos-001"

kernel = "/images/kernel/vmlinuz"

ramdisk = "/images/kernel/initrd.img"

extra = "ks=http://172.16.0.1/centos6.x86\_64.cfg"

memory = 512

vcpus = 2

vif = [ 'bridge=xenbr0' ]

disk = [ '/images/xen/centos6.6.img,qcow2,xvda,rw' ]

on\_reboot = "shutdown"

(4) 安装完成后，创建虚拟机的配置文件需要做出修改；

name = "centos-001"

bootloader = "pygrub"

memory = 512

vcpus = 2

vif = [ 'bridge=xenbr0' ]

disk = [ '/images/xen/centos6.6.img,qcow2,xvda,rw' ]

启动图形窗口：

在创建虚拟机的配置文件中定义vfb

(1) vfb = [ 'sdl=1' ]

(2) vnc

(a) yum install tigervnc

(b) vfb = [ 'vnc=1' ]

vnc监听的端口为5900, 相应的DISPLAYNUM为0

使用Dom0中物理磁盘分区为DomU提供存储空间：

使用libvirt实现xen虚拟机管理:

yum install libvirt libvirt-deamon-xen virt-manager python-virtinst libvirt-client

service libvirtd start

virt-manager, virsh, virt-install

实践：

(1) 提供DomU配置文件：

/etc/xen/bbox

(2) 创建磁盘映像文件，导致busybox文件系统，启动之；

补充：xm的配置文件：

kernel：内核

ramdisk: initramfs或initrd

name: 域名称

memory: 内存大小

disk: 磁盘设备文件列表，格式disk=["disk1", "disk2",], 每个disk都由三个参数进行定义：“backend-dev”，“frontend-dev”，“mode”

backend-dev: 有两种类型，物理设备，虚拟磁盘映像文件，格式为分别为phy:device和file:/path/to/image\_file;

front-dev: 定义其在DomU中设备类型；虚拟磁盘映像文件对应的设备文件名称通常为xvd[a-z]

mode: 访问权限模型，r, w

vcpus: 虚拟CPU的个数；

root: 根文件系统所在的设备；

extra: 传递给内核的额外参数；selinux=0

on\_reboot: 执行xm reboot命令时的操作，有destroy和restart;

on\_crash: 有destroy, restart, preserve(保存崩溃时的信息以用于调试)

vif ：vif = ['ip="172.16.100.11", bridge=br0']

type: 设备类型，默认为netfront

mac: 指定mac地址；

bridge: 指定桥接到的物理设备

ip: ip地址；

script: 配置此接口的脚本文件

vifname: 后端设备名称

bootloader: 引导器文件的路径，一般指的PyGrub的路径；

xm的配置文件说明

Xen配置文件一般由选项（options）、变量(variables)、CPU、网络、PCI、HVM、计时器(timers)、驱动(drivers)、磁盘设备(disk devices)、动作(behavior)，以及图形及声音(Graphics and audio)几个段组成，分别用于定义不同类别的域属性或设备属性。

上面的配置文件中的各选项作用如下。

◇ kernel：为当前域指定可用于DomU的内核文件；

◇ ramdisk：与kernel指定的内核文件匹配使用的ramdisk映像文件，根据需要指定，此为可选项；

◇ name：当前域的独有名称；每个域必须使用全局惟一的名称，否则将产生错误；

◇ memory：当前域的可用物理内存空间大小，单位为MB，默认为128；

◇ disk：当前域的所有可用磁盘设备列表，格式为disk = [ “disk1”, “disk2”, …]，每个disk都有三个参数进行定义，格式为“backend-dev，front-dev，mode”；

 backend-dev主要有两种类型，物理设备或虚拟磁盘映像文件，它们的格式分别为“phy:device”和“file:/path/to/file”；

 frontend-dev定义其在DomU中的设备类型，一般为xvd[a-z];

 mode则用于定义其访问权限，r为只读，w为读写；

◇ vcpus：配置给当前域使用的虚拟CPU的个数；默认为1；

◇ root：为当前域指定其根文件系统所在的设备，这个将作为内核参数在内核启动传递给内核；

◇ extra：传递给内核的额外参数，其中selinux=0表示禁用selinux，init则用于指定init程序的路径；多个参数之间使用空格隔开；

◇ on\_reboot：执行xm reboot命令或在当前域内部执行重启操作时由Xen执行的动作；其常用的值为destroy和restart；

◇ on\_crash：当前域由于各种原因崩溃时由Xen执行的动作；其常用的值为destroy、restart和preserve，preserve可以保存系统崩溃前的状态信息以用于调试；

◇ on\_shutdown：执行xm shutdown命令或在当前域内部执行关机操作时由Xen执行的动作；

/dev/sdb

disk = [ 'phy:/dev/sdb,xvda,w', ]

其它常用参数：

◇ vif：定义当前域的可用虚拟网络接口列表，每个虚拟网络接口都可以使用“name=value”的格式定义其属性；也可在定义某接口时不指定任何属性，其所有属性将均由系统默认配置；例如：vif = ['ip = "192.168.1.19", bridge=xenbr0']

 type：接口设备的类型，默认为netfront；

 mac：MAC地址，默认为随机；

 bridge：桥接到的物理设备，默认为Dom0中的第一个桥接设备；

 ip：ip地址；

 script：配置此接口的脚本文件，省略时将使用默认的配置脚本；

 vifname：后端设备的设备名称，默认为vifD.N，其中D为当前域的ID，N为此网络接口的ID；

◇ vfb：为当前域定义虚拟帧缓冲，其有许多可用属性，可以使用“name=value”的格式进行定义；

 vnc或sdl：定义vnc的类型，vnc=1表示启动一个可由外部设备连接的vnc服务器，sdl=1则表示启用一个自有的vncviewer；两者可以同时使用；

 vncdisplay：vnc显示号，默认为当前域的ID，当前域的VNC服务器将监听5900+此显示号的端口；

 vnclisten：VNC服务器监听的地址，默认为127.0.0.1；

 vncunused：如果此属性的值为非零值，则表示vncserver监听大于5900的第一个没被占用的端口；

 vncpasswd：指定VNC服务器的认证密码；

 display：用于域的自有vncviewer显示，默认为DISPLAY环境变量的值；

示例：

第一种：

vfb = [ 'sdl=1' ]

第二种：

vfb = [ 'vnc=1,vncpasswd=mageedu' ]

http://wiki.xen.org/wiki/XenConfigurationFileOptions#Graphics\_and\_Audio

◇ cpu：指定当前域应该在哪个物理CPU上启动，0表示第一颗CPU，1表示第二颗，依次类推；默认为-1，表示Xen可自行决定启动当前域的CPU；

◇ cpus：指定当前域的VCPU可以在哪些物理CPU上运行，如cpus = ”3,5-8,^6”表示当前域的VCPU可以在3，5，7，8号CPU上运行；

◇ bootloader：bootloader程序的路径；基于此bootloader，PV DomU的内核也可直接位于其文件系统上而非Dom0的文件系统；

更多的选项请参见xmdomain.cfg的手册页，或参考Xen官方wiki链接http://wiki.xen.org/wiki/XenConfigurationFileOptions中的详细解释。

在启动DomU时，可以为其定义可用的虚拟网络接口个数、每个虚拟网络接口的属性等，这仅需要在其对应的配置文件中使用vif选项即可。vif选项的值是一个列表，列表中的每个条目使用单引号引用，用于定义对应虚拟网络接口属性，条目之间使用逗号分隔。比如下面的示例就为当前域定义了三个虚拟网络接口，每个接口的属性均采用了默认配置。

vif = [ ‘ ‘, ‘ ‘, ‘ ‘ ]

每个网络接口可定义的属性语法格式为‘type= TYPE, mac=MAC, bridge=BRIDGE, ip=IPADDR, script= SCRIPT," + \ "backend=DOM, vifname=NAME, rate=RATE, model=MODEL, accel=ACCEL’。

◇ type：网络接口的类型，即其前端设备类型，默认为netfront；其可用的值还有有ioemu，表示使用QEMU的网络驱动；

◇ mac：指定此接口的MAC地址，默认为00:16:3E:(IEEE分配给XenSource的地址段)开头的随机地址；

◇ bridge：指定此接口使用的桥接设备，默认为Dom0内的第一个桥接设备；

◇ ip：为当前域定义固定IP地址，如果网络中存在DHCP服务器，请确保此地址一定没有包含于DHCP的可分配地址范围中；事实上，在DHCP环境中，可以直接在DHCP服务器上为此接口分配固定IP地址，因此，没有必要再使用此参数手动指定；

◇ script：指定用于配置当前接口网络属性的脚本，默认为xend的配置文件中使用vif-script指定的脚本；

◇ vifname：定义当前网络接口的后端设备在Dom0显示的名字；默认为vifDomID.DevID，其在当前域启动时自动生成，并随当前域ID的变化而改变；为其使用易于识别的固定名称有助于后期的管理工作；

◇ rate：为当前接口指定可用带宽，例如rate=10MB/s；

◇ model：由QEMU仿真的网络设备的类型，因此，只有在type的值为ioemu此参数才能意义；其可能的取值有lance、ne2k\_isa、ne2k\_pci、rt1839和smc91c111。默认为ne2k\_pci；

无论DomU中安装的是什么操作系统，为其定义网络接口时指定固定的MAC地址和接口名称通常是很有必要，因为其有助于追踪特定域的报文及当域多次启动后仍能使用相同的网络接口名称从而保证日志信息的连贯性。此外，如果Dom0中定义了多个桥接设备，还应该为桥接的网络接口使用bridge参数指定固定桥接到的桥接设备。下面的示例展示了指定此三个参数的接口定义。

vif = [ ‘vifname=web0.0, mac=00:16:3E:00:00:01, bridge=xenbr0’ ]

补充：Xen常见的工具栈

◇ Default / XEND

Xen 4.0及之前的版本中默认使用的工具栈，Xen 4.1提供了新的轻量级工具栈xl，但仍然保留了对Xend/xm的支持，但Xen 4.2及之后的版本已弃用。但xl在很大程度上保持了与xm的兼容。

◇ Default / XL

xl是基于libxenlight创建的轻量级命令行工具栈，并从Xen 4.1起成为默认的工具栈。xl与Xend的功能对比请参照http://wiki.xen.org/wiki/XL\_vs\_Xend\_Feature\_Comparison。

◇ XAPI / XE

XAPI即Xen管理API(The Xen management API)，它是Citrix XenServer和XCP默认使用的工具栈。目前，其移植向libxenlight的工作正进行中。XAPI是目前功能最通用且功能最完备的Xen工具栈，CloudStack、OpenNebula和OpenStack等云计算解决方案都基于此API管理Xen虚拟机。

补充资料：编译安装Xen

一、配置网络接口

[root@el6 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE="eth0"

HWADDR="00:11:22:33:44:55"

NM\_CONTROLLED="no"

ONBOOT="yes"

BOOTPROTO="dhcp"

最重要的是确保NM\_CONTROLLED的值为"no"，以及ONBOOT的值为"yes"。

另外，要确保网络服务能够开机自动启动：

# chkconfig network on

在/etc/hosts文件中为本机的主机名及IP地址建立解析

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.1.7 virtlab.magedu.com virtlab

禁用SELinux。

二、解决依赖关系

# yum install screen vim wget tcpdump ntp ntpdate man smartmontools links lynx ethtool xorg-x11-xauth

修改grub.conf，增加超时时间，并禁用hiddenmenu。

#boot=/dev/sda

default=0

timeout=15

splashimage=(hd0,0)/grub/splash.xpm.gz

#hiddenmenu

title Red Hat Enterprise Linux (2.6.32-279.el6.x86\_64)

　　　root (hd0,0)

　　　kernel /vmlinuz-2.6.32-279.el6.x86\_64 ro root=/dev/mapper/vg0-root rd\_NO\_LUKS LANG=en\_US.UTF-8 rd\_LVM\_LV=vg0/swap rd\_NO\_MD SYSFONT=latarcyrheb-sun16 crashkernel=auto rd\_NO\_DM KEYBOARDTYPE=pc KEYTABLE=us rd\_LVM\_LV=vg0/root rhgb quiet

　　　initrd /initramfs-2.6.32-279.el6.x86\_64.img

三、安装编译Xen所依赖的工具

# yum groupinstall "Development tools" "Additional Development" "Debugging Tools" "System administration tools" "Compatibility libraries" "Console internet tools" "Desktop Platform Development"

# yum install transfig wget texi2html libaio-devel dev86 glibc-devel e2fsprogs-devel gitk mkinitrd iasl xz-devel bzip2-devel pciutils-libs pciutils-devel SDL-devel libX11-devel gtk2-devel bridge-utils PyXML qemu-common qemu-img mercurial texinfo libuuid-devel ocaml ocaml-findlib

# yum install glibc-devel.i686

四、Rebuilding and installing Xen src.rpm

# cd /usr/local/src

# wget http://xenbits.xen.org/people/mayoung/EL6.xen/SRPMS/xen-4.1.2-20.el6.src.rpm

# rpm -ivh xen-4.1.2-20.el6.src.rpm

# cd /root/rpmbuild/SPECS/

# rpmbuild -bb xen.specs

五、安装qemu(此步非必须)

# yum install usbredir-devel spice-protocol spice-server-devel libseccomp-devel systemtap-sdt-devel nss-devel xfsprogs-devel bluez-libs-devel brlapi-devel libcap-devel

# wgethttp://mirrors.ustc.edu.cn/fedora/linux/releases/15/Everything/source/SRPMS/qemu-0.14.0-7.fc15.src.rpm

# rpm -ivh qemu-1.2.0-23.fc18.src.rpm

# cd /root/rpmbuild/SPECS

# rpmbuild -bb qemu.spec

六、编译内核

#

#

# These core options (Processor type and features| Paravirtualized guest support]

CONFIG\_PARAVIRT=y

CONFIG\_XEN=y

CONFIG\_PARAVIRT\_GUEST=y

CONFIG\_PARAVIRT\_SPINLOCKS=y

# add this item

# And Xen pv console device support (Device Drivers|Character devices

CONFIG\_HVC\_DRIVER=y

CONFIG\_HVC\_XEN=y

# And Xen disk and network support (Device Drivers|Block devices and Device Drivers|Network device support)

CONFIG\_XEN\_FBDEV\_FRONTEND=y

CONFIG\_XEN\_BLKDEV\_FRONTEND=y

# change the value to y

CONFIG\_XEN\_NETDEV\_FRONTEND=y

# change the value to y

# And the rest (Device Drivers|Xen driver support)

CONFIG\_XEN\_PCIDEV\_FRONTEND=y

CONFIG\_INPUT\_XEN\_KBDDEV\_FRONTEND=y

CONFIG\_XEN\_FBDEV\_FRONTEND=y

CONFIG\_XEN\_XENBUS\_FRONTEND=y

CONFIG\_XEN\_SAVE\_RESTORE=y

CONFIG\_XEN\_GRANT\_DEV\_ALLOC=m

# And for tmem support:

CONFIG\_XEN\_TMEM=y

# add the item

CONFIG\_CLEANCACHE=y

# enable the item

CONFIG\_FRONTSWAP=y

# enable the item

CONFIG\_XEN\_SELFBALLOONING=y

# add the item

# Configure kernel for dom0 support

# NOTE: Xen dom0 support depends on ACPI support. Make sure you enable ACPI support or you won't see Dom0 options at all.

# In addition to the config options above you also need to enable:

CONFIG\_X86\_IO\_APIC=y

CONFIG\_ACPI=y

CONFIG\_ACPI\_PROCFS=y (optional)

CONFIG\_XEN\_DOM0=y

CONFIG\_PCI\_XEN=y

CONFIG\_XEN\_DEV\_EVTCHN=y

CONFIG\_XENFS=y

# change the value to y

CONFIG\_XEN\_COMPAT\_XENFS=y

CONFIG\_XEN\_SYS\_HYPERVISOR=y

CONFIG\_XEN\_GNTDEV=y

CONFIG\_XEN\_BACKEND=y

CONFIG\_XEN\_NETDEV\_BACKEND=m

# enable the item

CONFIG\_XEN\_BLKDEV\_BACKEND=m

# enable the item

CONFIG\_XEN\_PCIDEV\_BACKEND=m

CONFIG\_XEN\_PRIVILEGED\_GUEST=y

CONFIG\_XEN\_BALLOON=y

CONFIG\_XEN\_SCRUB\_PAGES=y

# If you're using RHEL5 or CentOS5 as a dom0 (ie. you have old udev version), make sure you enable the following options as well:

CONFIG\_SYSFS\_DEPRECATED=y

CONFIG\_SYSFS\_DEPRECATED\_V2=y

七、编辑安装libvirt

yum install libblkid-devel augeas sanlock-devel radvd ebtables systemtap-sdt-devel scrub numad

# cd /usr/src

# wget ftp://ftp.redhat.com/pub/redhat/linux/enterprise/6Server/en/os/SRPMS/libvirt-0.9.10-21.el6\_3.7.src.rpm

# rpm -ivh libvirt-0.9.10-21.el6\_3.7.src.rpm

编辑/root/rpmbuild/SPECS/libvirt.spec文件，启用with\_xen选项。

# cd /root/rpmbuild/SPECS

# rpmbuild -bb libvirt.spec

yum install libnl-devel xhtml1-dtds libudev-devel libpciaccess-devel yajl-devel libpcap-devel avahi-devel parted-devel device-mapper-devel numactl-devel netcf-devel xen-devel dnsmasq iscsi-initiator-utils gtk-vnc-python

先安装gtk-vnc-python包。

编译安装virtinst

编译安装libvirtd

编译安装virt-manager

注意：libvirt-1.0之前的版本不支持xen 4.2。

补充资料：杂项

桥接功能是在内核中实现的，因此，只要提供了正确网络接口的配置文件，Linux系统会自动启用桥接设备。比如，如果想为当前系统定义桥接设备xenbr0，并将eth0设备附加其上，在rhel6平台上，其步骤如下。

假设eth0的原始配置信息如下面所示。

DEVICE="eth0"

BOOTPROTO="static"

NM\_CONTROLLED="no"

HWADDR=00:0C:29:B4:48:BB

IPADDR=192.168.1.7

NETMASK=255.255.255.0

GATEWAY=192.168.1.1

ONBOOT="yes"

TYPE="Ethernet"

为xenbr0创建配置文件/etc/sysconfig/network-scripts/ifcfg-xenbr0，其内容类似下面所示。需要注间的是TYPE属性的值Bridge一定要大写首字母。

DEVICE="xenbr0"

BOOTPROTO="static"

NM\_CONTROLLED="no"

IPADDR=192.168.1.7

NETMASK=255.255.255.0

GATEWAY=192.168.1.1

ONBOOT="yes"

TYPE=" Bridge"

DELAY=0

将eth0的配置文件/etc/sysconfig/network-scripts/ifcfg-eth0的内容修改为：

DEVICE="eth0"

BOOTPROTO="static"

NM\_CONTROLLED="no"

HWADDR=00:0C:29:B4:48:BB

ONBOOT="yes"

TYPE="Ethernet"

BRIDGE=xenbr0

而后使用/etc/init.d/network脚本重启网络服务即可。

使用brctl命令管理桥接设备

brctl是由bridge-utils软件包提供的命令，使用前请先确保其已经正确安装。brctl有许多的子命令，分别用于实现添加、删除桥接设备，向桥设备附加网络接口等。不带任何参数的brctl命令即可显示其使用帮助，如下列出了其中的一部分。

# brctl

Usage: brctl [commands]

commands:

addbr <bridge> add bridge

delbr <bridge> delete bridge

addif <bridge> <device> add interface to bridge

delif <bridge> <device> delete interface from bridge

show show a list of bridges

…… ……

由上面的帮助信息所示，添加一个桥接设备只需使用命令“brctl addbr <bridge\_name>”即可，比如，添加xenbr0设备：

# brctl addbr xenbr0

刚添加桥接设备时，由于没有为其附加任何网络接口，其尚处于未激活状态。为其附加一个网络接口设备可使用“brctl addif <bridge\_name> <device\_name>命令实现，比如，将eth0附加至刚添加的xenbr0：

# brctl addif xenbr0 eth0

从桥设备上删除某网络接口，使用命令“brctl delif <bridge\_name> <device\_name>”；删除桥接设备则使用“brctl delbr <bridge\_name>”，不过需要注意的是，只有在桥设备处于未激活状态时才能将其删除。而要查看桥接设备相关的信息，可以使用“brctl show”命令。

需要注意的是，使用brctl命令配置桥接设备的完整过程需要参考前面讲解的network-bridge start命令的实现步骤进行。

# To create one using the VNC backend and sensible defaults:

#

# vfb = [ 'type=vnc' ]

#

# The backend listens on 127.0.0.1 port 5900+N by default, where N is

# the domain ID. You can override both address and N:

#

# vfb = [ 'type=vnc,vnclisten=127.0.0.1,vncdisplay=1' ]

#

# Or you can bind the first unused port above 5900:

#

# vfb = [ 'type=vnc,vnclisten=0.0.0.0,vncunused=1' ]

#

# You can override the password:

#

# vfb = [ 'type=vnc,vncpasswd=MYPASSWD' ]

#

# Empty password disables authentication. Defaults to the vncpasswd

# configured in xend-config.sxp.

vfb = [ 'type=vnc,vncdisplay=10,vncpasswd=s3cr3t' ]

批量部署DomU:

准备一个映像模板；

OZ：

脚本：

生成一个配置文件/etc/xen

下载一个磁盘映像

16：

disk = ['phy:/dev/sdb,xvda,w']

使用了bootloader, pygrup示例：

#ramdisk="/boot/initramfs-2.6.32-358.el6.x86\_64.img"

#kernel="/boot/vmlinuz-2.6.32-358.el6.x86\_64"

name="linux"

vcpus=1

memory=128

disk=['file:/xen/vm2/dom2.img,xvda,w',]

bootloader="/usr/bin/pygrub"

#root="/dev/xvda2 ro"

#extra="selinux=0 init=/sbin/init"

vif=[ 'bridge=br0' ]

on\_crash="destroy"

on\_reboot="restart"

使用Dom0中的kernel和ramdisk引导的示例：

ramdisk="/boot/initramfs-2.6.32-358.el6.x86\_64.img"

kernel="/boot/vmlinuz-2.6.32-358.el6.x86\_64"

name="test"

vcpus=1

memory=128

disk=['file:/xen/vm1/test.img,xvda,w',]

root="/dev/xvda ro"

extra="selinux=0 init=/sbin/init"

自定义安装，并启用了vnc功能：

#ramdisk="/xen/isolinux/initrd.img"

#kernel="/xen/isolinux/vmlinuz"

name="rhel6"

vcpus=2

memory=512

disk=['file:/xen/vm3/rhel6.img,xvda,w',]

bootloader="/usr/bin/pygrub"

#root="/dev/xvda2 ro"

#extra="selinux=0 init=/sbin/init"

#extra="ks=http://172.16.0.1/rhel6.x86\_64.cfg"

vif=[ 'bridge=br0' ]

on\_crash="destroy"

on\_reboot="destroy"

vfb=[ 'vnc=1,vnclisten=0.0.0.0' ]

补充资料：创建Xen PV模式虚拟机的前提

在PV模式中运行guest系统，需要满足几个基本前提。

◇ 能运行于Xen DomU的(Xen-enabled)内核：Liunx 2.6.24及以后的内核已经添加了对Xen DomU的支持，因此，只要在内核编译时启用了相应的功能就能满足此要求，目前多数Linux发行版的内核都已经支持此特性；而此前的版本需要内核在编译前手动打补丁方可；

◇ 根文件系统(Root Filesystem)：包含了应用程序、系统组件及配置文件等运行DomU的各种所需要文件的文件系统，其不用非得包含内核及对应的ramdisk，后面的这些组件放在Dom0中即可；事实上，用于DomU的内核文件必须要能够允许Dom0访问到，因为其运行时需要与Xen Hypervisor通信，因此，这些内核组件可以位于Dom0能够访问到的任何文件系统上；然而，目前基于pygrub(可用于Dom0跟非特权域磁盘映像中的内核通信)，此内核文件也可以直接放置于非特权域的磁盘映像中；

◇ DomU内核所需要的内核模块：内核模块是内核的重要组成部分，它们一般存储于根文件系统；

◇ ramdisk或者ramfs：这个根据实际需要是个可选组件，如果在内核初始化过程中不需要依赖于此来装载额外的驱动程序以访问根文件系统则也可以不用提供；

◇ swap设备：交换分区能够让Linux运行比仅有物理内存时更多的进程，因此，提供此组件是常见的做法；当然，它是可选的；

◇ DomU配置文件：集中在一起指定前述各组件的配置信息，以及定义其它有关PV DomU的基本属性的文件；其通常包含所有用于当前DomU属性配置参数，包括为其指定磁盘映像和内核文件的位置(或pygrub的位置)等，以及其它许多属性如当前DomU可以访问的设备等，这些设备包括网络设备、硬盘、显卡及其它PCI设备；同时，配置文件中也可以指定新创建的非特权域可以使用的物理内存大小及虚拟CPU个数等等；

这里需要提醒的是，如果计划为PV DomU编译内核，需要以与传统方式不同的方式放置内核及其模块。前面也已经提到，内核一般会放在Dom0的某路径下，而内核模块则需要放在DomU的根文件系统上。

PV DomU的根文件系统可以以多种不同的方式进行安置，比如：

◇ 虚拟磁盘映像文件

◇ Dom0没有使用的额外物理磁盘分区

◇ Dom0没有使用的逻辑卷

◇ 块级别网络文件系统，如iSCSI设备

分布式文件系统

◇ 网络文件系统，如NFS

有许多组织提供了预配置的根文件系统，如FreeOsZoo(www.oszoo.org)、Jailtime.org(www.jailtime.org)等，读者可以根据需要到它们的站点下载。另外，rPath还提供了一个在的根文件系统制作系统rBuilder(www.rpath.com/rbuiler)。

补充资料：基于isolinux的内核和initrd文件在PV DomU模式安装CentOS 6.6

第一步，创建空的虚拟磁盘映像，以之作为新建非特权域的虚拟磁盘文件，此映像文件并不真正占用为其指定的空间，而是随着存储的内容而变化。

# qemu-img create -f qcow2 -o size=120G,preallocation=metadata /xen/images/centos6.img

第二步，获取要安装的CentOS6(x86\_64)的isolinux目录中的vmlinuz和initrd.img文件，这里将其存放于/xen/centos6/isolinux目录中。

第三步，为新的非特权域创建配置文件/etc/xen/centos6，内容如下。

kernel = "/xen/rhel6/isolinux/vmlinuz"

ramdisk = "/xen/rhel6/isolinux/initrd.img"

name = "centos6"

memory = "512"

disk = [ 'file:/xen/iamges/centos6.img,xvda,w', ]

vif = [ 'bridge=xenbr0', ]

#bootloader="/usr/bin/pygrub"

#extra = "text ks=http://some\_server/path/to/kickstart.cfg"

vcpus=1

on\_reboot = 'destroy'

on\_crash = 'destroy'

第四步，创建新的非特权域。

# xm create -c centos6

此命令会启动rhel6的安装界面，其运行于Xen的PV模式。根据提示一步步的安装系统即可。

安装完成后，将上述配置文件中的前两行信息注释，并启用bootloader一行，再使用如前面的xm create命令即可正常启动此域。其修改后的配置文件内容如下：

name = "centos6"

memory = "512"

disk = [ 'file:/xen/images/centos6.img,xvda,w', ]

vif = [ 'bridge=xenbr0', ]

bootloader="/usr/bin/pygrub"

vcpus=1

on\_reboot = 'restart'

on\_crash = 'destroy

xen实时迁移

基于iscsi来实现。

kernel = "/boot/vmlinuz-3.7.4-1.el6xen.x86\_64"

ramdisk = "/boot/initramfs-3.7.4-1.el6xen.x86\_64.img"

name = "test"

memory = "128"

disk = [ 'phy:/dev/sdb,xvda,w', ]

vcpus=2

on\_reboot = 'restart'

on\_crash = 'destroy'

root = "/dev/xvda2 ro"

extra = "selinux=0 init=/sbin/init"

提供配置文件：

/etc/{passwd,shadow,group,fstab,nsswitch.conf}

提供库文件：

/lib64/libnss\_file.so.\*

提供配置文件:/etc/nginx

提供目录：/var/run, /var/log/nginx

网卡驱动：xen-netfront.ko

[xen4]

name=Xen4 Project

baseurl=ftp://172.16.0.1/pub/Sources/6.x86\_64/xen4/x86\_64/

gpgcheck=0

cost=500

xen实时迁移，xm, 配置文件的使用细节

[et-virt07 ~]# grep xend-relocation /etc/xen/xend-config.sxp |grep -v '#'

(xend-relocation-server yes)

(xend-relocation-port 8002)

(xend-relocation-address '')

(xend-relocation-hosts-allow '')

[et-virt08 ~]# grep xend-relocation /etc/xen/xend-config.sxp |grep -v '#'

(xend-relocation-server yes)

(xend-relocation-port 8002)

(xend-relocation-address '')

(xend-relocation-hosts-allow '')

KVM:

虚拟化技术有两种类型的实现：

Type-I:

hypervisor --> vm

Type-II:

host --> vmm --> vms

Xen：

hypervisor, Dom0

KVM: Kernel-based Virtual Machine, Qumranet公司, 依赖于HVM；Intel VT-x, ADM ADM-V；

KVM模块载入后的系统的运行模式：

内核模式：GuestOS执行I/O类操作，或其它的特殊指令的操作；称作“来宾-内核”模式；

用户模式：代表GuestOS请求I/O类操作；

来宾模式：GuestOS的非I/O类操作；事实上，它被称作“来宾-用户”模式；

kvm hypervisor：

KVM的组件：

两类组件：

/dev/kvm：工作于hypervisor，在用户空间可通过ioctl()系统调用来完成VM创建、启动等管理功能；它是一个字符设备，功能：

创建VM、为VM分配内存、读写VCPU的寄存器、向VCPU注入中断、运行VCPU等等；

qemu进程：工作于用户空间，主要用于实现模拟PC机的IO设备；

KVM特性：

内存管理：

将分配给VM的内存交换至SWAP；

支持使用Huge Page;

支持使用Intel EPT或AMD RVI技术完成内存地址映射；GVA-->GPA-->HPA

支持KSM (Kernel Same-page Merging)

硬件支持：

取决于Linux内核；

存储：

本地存储：

网络附加存储：

存储区域网络：

分布式存储：例如GlustFS

实时迁移：

支持的GuestOS:

Linux, Windows, OpenBSD, FreeBSD, OpenSolaris;

设备驱动：

IO设备的完全虚拟化：模拟硬件

IO设备的半虚拟化：在GuestOS中安装驱动；virtio

virtio-blk, virtio-net, virtio-pci, virtio-console, virtio-ballon

KVM局限性：

一般局限性：

CPU overcommit

时间记录难以精确，依赖于时间同步机制

MAC地址：

VM量特别大时，存在冲突的可能性；

实时迁移：

性能局限性：

KVM的工具栈：

qemu：

qemu-kvm

qemu-img

libvirt

GUI: virt-manager, virt-viewer

CLI: virt-install, virsh

QEMU主要提供了以下几个部分：

处理器模拟器

仿真IO设备

关联模拟的设备至真实设备；

调试器

与模拟器交互的用户接口

安装：

(1) 确保CPU支持HVM

# grep -E --color=auto "(vmx|svm)" /proc/cpuinfo

(2) 装载模块

# modprobe kvm

# modprobe kvm-intel

(3) 验正：

/dev/kvm

管理工具栈：

# yum grouplist | grep -i "virtualization"

Virtualization：提供qemu-kvm

Virtualization Client:提供python-virtinst, virt-manager, virt-viewer

Virtualization Platform:提供libvirt, libvirt-client

Virtualization Tools:提供 libguestfs

补充资料：KVM内存管理

KVM继承了Linux系统管理内存的诸多特性，比如，分配给虚拟使用的内存可以被交换至交换空间、能够使用大内存页以实现更好的性能，以及对NUMA的支持能够让虚拟机高效访问更大的内存空间等。

KVM基于Intel的EPT（Extended Page Table）或AMD的RVI（Rapid Virtualization Indexing）技术可以支持更新的内存虚拟功能，这可以降低CPU的占用率，并提供较好的吞吐量。

此外，KVM还借助于KSM（Kernel Same-page Merging）这个内核特性实现了内存页面共享。KSM通过扫描每个虚拟机的内存查找各虚拟机间相同的内存页，并将这些内存页合并为一个被各相关虚拟机共享的单独页面。在某虚拟机试图修改此页面中的数据时，KSM会重新为其提供一个新的页面副本。实践中，运行于同一台物理主机上的具有相同GuestOS的虚拟机之间出现相同内存页面的概率是很的，比如共享库、内核或其它内存对象等都有可能表现为相同的内存页，因此，KSM技术可以降低内存占用进而提高整体性能。

补充资料：

VMM:对IO的驱动有三种模式：

自主VMM：VMM自行提供驱动和控制台；

混合VMM：借助于OS提供驱动；

依赖于外部OS实现特权域

自我提供特权域

寄宿式VMM：

IO虚拟化模型：

模拟

半虚拟化

透传

使用qemu-kvm管理工具：

yum install qemu-kvm

/usr/libexec/qemu-kvm

命令选项：

标准选项：

显示选项：

　　　块设备选项：

　　　网络选项：

i386平台专用选项

字符设备选项

蓝牙设备选项

Linux启动专用选项

调试/专家模式选项

cirros project: 为cloud环境测试vm提供的微缩版Linux；

启动第一个虚拟：

qemu-kvm -m 128 -smp 2 -name test -hda /images/kvm/cirros-0.3.4-i386.disk.img

用-drive指定磁盘映像文件：更高级

# qemu-kvm -m 128 -name test -smp 2 -drive file=/images/kvm/cirros-0.3.4-i386-disk.img,if=virtio,media=disk,cache=writeback,format=qcow2

通过cdrom启动winxp的安装：

# qemu-kvm -name winxp -smp 4,sockets=1,cores=2,threads=2 -m 512 -drive file=/images/kvm/winxp.img,if=ide,media=disk,cache=writeback,format=qcow2 -drive file=/root/winxp\_ghost.iso,media=cdrom

指定使用桥接网络接口：

qemu-kvm -m 128 -name test -smp 2 -drive file=/images/kvm/cirros-0.3.4-i386-disk.img,if=virtio,media=disk,cache=writeback,format=qcow2 -net nic -net tap,script=/etc/if-up,downscript=no -nographic

显示选项：

SDL: Simple DirectMedia Layer：C语言开发，跨平台且开源多媒体程序库文件；

在qemu中使用“-sdl”即可；

VNC: Virtual Network Computing，使用RFB(Remote FrameBuffer)协议远程控制另外的主机；

CentOS 6.6

(1) yum install tigervnc-server

(2) vncpasswd

(3) vncserver :N

qemu-kvm

-vnc display,option,option

示例：-vnc :N,password

启动qemu-kvm时，额外使用-monitor stdio选项，并使用

change vnc password命令设置密码；

补充资料：qemu-kum使用文档

2.5.6 使用qemu-kvm管理KVM虚拟机

Qemu是一个广泛使用的开源计算机仿真器和虚拟机。当作为仿真器时，可以在一种架构(如PC机)下运行另一种架构(如ARM)下的操作系统和程序。而通过动态转化，其可以获得很高的运行效率。当作为一个虚拟机时，qemu可以通过直接使用真机的系统资源，让虚拟系统能够获得接近于物理机的性能表现。qemu支持xen或者kvm模式下的虚拟化。当用kvm时，qemu可以虚拟x86、服务器和嵌入式powerpc，以及s390的系统。

QEMU 当运行与主机架构相同的目标架构时可以使用 KVM。例如，当在一个x86兼容处理器上运行 qemu-system-x86 时，可以利用 KVM 加速——为宿主机和客户机提供更好的性能。

Qemu有如下几个部分组成：

◇ 处理器模拟器(x86、PowerPC和Sparc)；

◇ 仿真设备(显卡、网卡、硬盘、鼠标等)；

◇ 用于将仿真设备连接至主机设备(真实设备)的通用设备；

◇ 模拟机的描述信息；

◇ 调试器；

◇ 与模拟器交互的用户接口；

2.5.6.1 使用qemu-kvm安装Guest

如2.5.5中所述，基于libvirt的工具如virt-manager和virt-install提供了非常便捷的虚拟机管理接口，但它们事实上上经二次开发后又封装了qemu-kvm的工具。因此，直接使用qemu-kvm命令也能够完成此前的任务。

2.5.6.1.1 qemu-kvm命令

在RHEL6上，qemu-kvm位于/usr/libexec目录中。由于此目录不属于PATH环境变量，故无法直接使用，这样也阻止了可以直接使用qemu作为创建并管理虚拟机。如若想使用qemu虚拟机，可以通过将/usr/libexec/qemu-kvm链接为/usr/bin/qemu实现。

# ln -sv /usr/lib/exec/qemu-kvm /usr/bin/qemu-kvm

qemu-kvm命令使用格式为“qemu-kvm [options] [disk\_image]”，其选项非常多，不过，大致可分为如下几类。

◇ 标准选项；

◇ USB选项；

◇ 显示选项；

◇ i386平台专用选项；

◇ 网络选项；

◇ 字符设备选项；

◇ 蓝牙相关选项；

◇ Linux系统引导专用选项；

◇ 调试/专家模式选项；

◇ PowerPC专用选项；

◇ Sparc32专用选项；

考虑到篇幅及使用需要，这里介绍的选项主要涉及到标准选项、显示选项、i386平台专用选项及Linux系统引导专用选项等相关的选项。

2.5.6.1.2 qemu-kvm的标准选项

qemu-kvm的标准选项主要涉及指定主机类型、CPU模式、NUMA、软驱设备、光驱设备及硬件设备等。

◇ -name name：设定虚拟机名称；

◇ -M machine：指定要模拟的主机类型，如Standard PC、ISA-only PC或Intel-Mac等，可以使用“qemu-kvm -M ?”获取所支持的所有类型；

◇ -m megs：设定虚拟机的RAM大小；

◇ -cpu model：设定CPU模型，如coreduo、qemu64等，可以使用“qemu-kvm -cpu ?”获取所支持的所有模型；

◇ -smp n[,cores=cores][,threads=threads][,sockets=sockets][,maxcpus=maxcpus]：设定模拟的SMP架构中CPU的个数等、每个CPU的核心数及CPU的socket数目等；PC机上最多可以模拟255颗CPU；maxcpus用于指定热插入的CPU个数上限；

◇ -numa opts：指定模拟多节点的numa设备；

◇ -fda file

◇ -fdb file：使用指定文件(file)作为软盘镜像，file为/dev/fd0表示使用物理软驱；

◇ -hda file

◇ -hdb file

◇ -hdc file

◇ -hdd file：使用指定file作为硬盘镜像；

◇ -cdrom file：使用指定file作为CD-ROM镜像，需要注意的是-cdrom和-hdc不能同时使用；将file指定为/dev/cdrom可以直接使用物理光驱；

◇ -drive option[,option[,option[,...]]]：定义一个硬盘设备；可用子选项有很多。

 file=/path/to/somefile：硬件映像文件路径；

 if=interface：指定硬盘设备所连接的接口类型，即控制器类型，如ide、scsi、sd、mtd、floppy、pflash及virtio等；

 index=index：设定同一种控制器类型中不同设备的索引号，即标识号；

 media=media：定义介质类型为硬盘(disk)还是光盘(cdrom)；

 snapshot=snapshot：指定当前硬盘设备是否支持快照功能：on或off；

 cache=cache：定义如何使用物理机缓存来访问块数据，其可用值有none、writeback、unsafe和writethrough四个；

 format=format：指定映像文件的格式，具体格式可参见qemu-img命令；

◇ -boot [order=drives][,once=drives][,menu=on|off]：定义启动设备的引导次序，每种设备使用一个字符表示；不同的架构所支持的设备及其表示字符不尽相同，在x86 PC架构上，a、b表示软驱、c表示第一块硬盘，d表示第一个光驱设备，n-p表示网络适配器；默认为硬盘设备；

-boot order=dc,once=d

2.5.6.1.3 qemu-kvm的显示选项

显示选项用于定义虚拟机启动后的显示接口相关类型及属性等。

◇ -nographic：默认情况下，qemu使用SDL来显示VGA输出；而此选项用于禁止图形接口，此时,qemu类似一个简单的命令行程序，其仿真串口设备将被重定向到控制台；

◇ -curses：禁止图形接口，并使用curses/ncurses作为交互接口；

◇ -alt-grab：使用Ctrl+Alt+Shift组合键释放鼠标；

◇ -ctrl-grab：使用右Ctrl键释放鼠标；

◇ -sdl：启用SDL；

◇ -spice option[,option[,...]]：启用spice远程桌面协议；其有许多子选项，具体请参照qemu-kvm的手册；

◇ -vga type：指定要仿真的VGA接口类型，常见类型有：

 cirrus：Cirrus Logic GD5446显示卡；

 std：带有Bochs VBI扩展的标准VGA显示卡；

 vmware：VMWare SVGA-II兼容的显示适配器；

 qxl：QXL半虚拟化显示卡；与VGA兼容；在Guest中安装qxl驱动后能以很好的方式工作，在使用spice协议时推荐使用此类型；

 none：禁用VGA卡；

◇ -vnc display[,option[,option[,...]]]：默认情况下，qemu使用SDL显示VGA输出；使用-vnc选项，可以让qemu监听在VNC上，并将VGA输出重定向至VNC会话；使用此选项时，必须使用-k选项指定键盘布局类型；其有许多子选项，具体请参照qemu-kvm的手册；

display:

（1）host:N

172.16.100.7:1, 监听于172.16.100.7主的5900+N的端口上

(2) unix:/path/to/socket\_file

(3) none

options:

password: 连接时需要验正密码；设定密码通过monitor接口使用change

reverse: “反向”连接至某处于监听状态的vncview上；

-monitor stdio：表示在标准输入输出上显示monitor界面

-nographic

Ctrl-a, c: 在console和monitor之间切换

Ctrl-a, h: 显示帮助信息

2.5.6.1.4 i386平台专用选项

◇ -no-acpi：禁用ACPI功能，GuestOS与ACPI出现兼容问题时使用此选项；

◇ -balloon none：禁用balloon设备；

◇ -balloon virtio[,addr=addr]：启用virtio balloon设备；

2.5.6.1.5 网络属性相关选项

网络属性相关选项用于定义网络设备接口类型及其相关的各属性等信息。这里只介绍nic、tap和user三种类型网络接口的属性，其它类型请参照qemu-kvm手册。

◇ -net nic[,vlan=n][,macaddr=mac][,model=type][,name=name][,addr=addr][,vectors=v]：创建一个新的网卡设备并连接至vlan n中；PC架构上默认的NIC为e1000，macaddr用于为其指定MAC地址，name用于指定一个在监控时显示的网上设备名称；emu可以模拟多个类型的网卡设备，如virtio、i82551、i82557b、i82559er、ne2k\_isa、pcnet、rtl8139、e1000、smc91c111、lance及mcf\_fec等；不过，不同平台架构上，其支持的类型可能只包含前述列表的一部分，可以使用“qemu-kvm -net nic,model=?”来获取当前平台支持的类型；

　　　注意：默认情况mac地址会自动分配为

◇ -net tap[,vlan=n][,name=name][,fd=h][,ifname=name][,script=file][,downscript=dfile]：通过物理机的TAP网络接口连接至vlan n中，使用script=file指定的脚本(默认为/etc/qemu-ifup)来配置当前网络接口，并使用downscript=file指定的脚本(默认为/etc/qemu-ifdown)来撤消接口配置；使用script=no和downscript=no可分别用来禁止执行脚本；

◇ -net user[,option][,option][,...]：在用户模式配置网络栈，其不依赖于管理权限；有效选项有：

 vlan=n：连接至vlan n，默认n=0；

 name=name：指定接口的显示名称，常用于监控模式中；

 net=addr[/mask]：设定GuestOS可见的IP网络，掩码可选，默认为10.0.2.0/8；

 host=addr：指定GuestOS中看到的物理机的IP地址，默认为指定网络中的第二个，即x.x.x.2；

 dhcpstart=addr：指定DHCP服务地址池中16个地址的起始IP，默认为第16个至第31个，即x.x.x.16-x.x.x.31；

 dns=addr：指定GuestOS可见的dns服务器地址；默认为GuestOS网络中的第三个地址，即x.x.x.3；

 tftp=dir：激活内置的tftp服务器，并使用指定的dir作为tftp服务器的默认根目录；

 bootfile=file：BOOTP文件名称，用于实现网络引导GuestOS；如：qemu -hda linux.img -boot n -net user,tftp=/tftpserver/pub,bootfile=/pxelinux.0

# cat /etc/qemu-ifup

#!/bin/bash

#

bridge=br0

　　　#kvm自动将后半段传递过来

#br0如果不存在还得手动加上去

if [ -n "$1" ]; then

ip link set $1 up

sleep 1

brctl addif $bridge $1

[ $? -eq 0 ] && exit 0 || exit 1

else

echo "Error: no interface specified."

exit 1

fi

# cat /etc/qemu-ifdown

#!/bin/bash

#

bridge=br0

if [ -n "$1" ];then

brctl delif $bridge $1

ip link set $1 down

exit 0

else

echo "Error: no interface specified."

exit 1

fi

2.5.6.1.6 一个使用示例

下面的命令创建了一个名为rhel5.8的虚拟机，其RAM大小为512MB，有两颗CPU的SMP架构，默认引导设备为硬盘，有一个硬盘设备和一个光驱设备，网络接口类型为virtio，VGA模式为cirrus，并启用了balloon功能。

# qemu-kvm -name "rhel5.8" -m 512 \

-smp 2 -boot d \

-drive file=/VM/images/rhel5.8/hda,if=virtio,index=0,media=disk,format=qcow2 \

-drive file=/isos/rhel-5.8.iso,index=1,media=cdrom \

-net nic,model=virtio,macaddr=52:54:00:A5:41:1E \

-vga cirrus -balloon virtio

需要注意的是，上述命令中使用的硬盘映像文件/VM/images/rhel5.8/hda需要事先使用qemu-img命令创建，其具体使用格式请见下节介绍。

在虚拟机创建并安装GuestOS完成之后，可以免去光驱设备直接启动之。命令如下所示。

# qemu-kvm -name "rhel5.8" -m 512 \

-smp 2 -boot d \

-drive file=/VM/images/rhel5.8/hda,if=virtio,index=0,media=disk,format=qcow2 \

-net nic,model=virtio,macaddr=52:54:00:A5:41:1E \

-vga cirrus -balloon virtio

2.5.6.1.7 使用qemu-img管理磁盘映像

qemu-img是qemu用来实现磁盘映像管理的工具组件，其有许多子命令，分别用于实现不同的管理功能，而每一个子命令也都有一系列不同的选项。其使用语法格式为“qemu-img subcommand [options]”，支持的子命令如下。

◇ create：创建一个新的磁盘映像文件；

◇ check：检查磁盘映像文件中的错误；

◇ convert：转换磁盘映像的格式；

◇ info：显示指定磁盘映像的信息；

◇ snapshot：管理磁盘映像的快照；

◇ commit：提交磁盘映像的所有改变；

◇ rbase：基于某磁盘映像创建新的映像文件；

◇ resize：增大或缩减磁盘映像文件的大小；

使用create子命令创建磁盘映像的命令格式为“create [-f fmt] [-o options] filename [size]”，例如下面的命令创建了一个格式为qcow2的120G的稀疏磁盘映像文件。

# qemu-img create -f qcow2 /VM/images/rhel5.8/hda 120G

Formatting '/VM/images/rhel5.8/hda', fmt=qcow2 size=128849018880 encryption=off cluster\_size=65536

更进一步的使用信息请参照手册页。

KVM(2)

KVM的网络功能

qemu-kvm所提供的网络模式：

基于网桥的虚拟网卡；-net tap

基于NAT的虚拟网络；

Qemu内置的用户网络模式；-net user

直接分配网络设备(VT-d, SR-IOV)

-net nic：frontend, 为VM添加虚拟网卡并指明虚拟网卡特性

-net user, -net tap: backend, 定义虚拟网络，并指定如何将VM的虚拟网卡连入虚拟网络

-net none: 禁用vm的网络功能

-net nic -net tap, -net nic -net user

-net nic[,vlan=n][,macaddr=mac][,model=type][,name=name][,addr=addr][,vectors=v]

-net nic,model=virtio

查看本机的qemu-kvm支持网络接口类型：

# qemu-kvm -net nic,model=?

qemu: Supported NIC models: ne2k\_pci,i82551,i82557b,i82559er,rtl8139,e1000,pcnet,virtio

注意：(1) 如果需要为VM添加多块网卡，则要多使用“-net nic”选项；

(2) 需要为VM的网卡指定MAC地址，地址范围属于“52:54:00”开头的地址块；

-net tap[,vlan=n][,name=name][,fd=h][,ifname=name][,script=file][,downscript=dfile]

ifname=

script=/path/to/some\_script：虚拟机启动时，tap为其创建的Nic的后半段会保留在host上，在host之上通常需要将其添加至某桥上，实现虚拟网络功能；

downscript=/path/to/some\_script: 虚拟机关闭时，如果处理此前的启动脚本为其设置网络；

kvm常用的虚拟网络模型：

桥接模型

NAT模型

路由模型

隔离模型

手动设置接口添加至指定桥的过程：

brctl addbr $BR

ip link set $IF up

ip link set $BR up

brctl addif $BR $IF

ip link set $BR promisc on

virtio半虚拟化：

HVM：虚拟化CPU

I/O半虚拟化分成两段：

前端驱动(virtio前半段)：virtio-blk, virtio-net, virtio-pci, virtio-balloon, virtio-console

Linux：CentOS 4.8+, 5.3+, 6.0+, 7.0+

Windows：

virtio: 虚拟队列，virt-ring（环空间占用完进入等待）

transport：

后端处理程序(virt backend drivers)：在QEMU中实现；

virtio-balloon: 像气球一样弹性扩大或缩小虚拟机使用的空间

ballooning: 让VM中运行的GuestOS中运行调整其内存大小；

# qemu-kvm -balloon virtio

手动查看GuestOS的内存用量：

info balloon

balloon N

virtio-net：

其依赖于GuestOS中的驱动，及Qemu中的后端驱动

GuestOS: virtio\_net.ko

Qemu: qemu-kvm -net nic,model=?

qemu-kvm -net nic,model=virtio

Host中的GSO, TSO

关掉可能会提升性能：

ethtool -K $IF gso off

ethtool -K $IF tso off

ethtool -k $IF

vhost-net：用于取代工作于用户空间的qemu中为virtio-net实现的后端驱动以实现性能提升的驱动；

-net tap[,vnet\_hdr=on|off][,vhost=on|off]

qemu-kvm -net tap,vnet\_hdr=on,vhost=on

virtio-blk：

其依赖于GuestOS中的驱动，及Qemu中的后端驱动

-drive file=/path/to/some\_image\_file,if=virtio

kvm\_clock: 半虚拟化的时钟

# grep -i "paravirt" /boot/config-2.6.32-504.el6.x86\_64

CONFIG\_PARAVIRT\_GUEST=y

CONFIG\_PARAVIRT=y

CONFIG\_PARAVIRT\_CLOCK=y

VM Migration：

static migration

live migration

整体迁移时间

服务器停机时间

对服务的性能的影响

在待迁入主机使用

# qemu-kvm -vnc :N -incoming tcp:0:7777

# vncviewer :590N

在源主机使用：

monitor接口：

migrate tcp:DEST\_IP:DEST:PORT

补充资料：nat模型网络脚本示例：

亲测实验：

创建实例

\*\*\*\*\*经过测试如果-net没有指定model=virtio，默认用的e1000，那虚拟机的网络十分不稳定或者无法使用

[root@localhost ~]# qemu-kvm -m 1G -smp 2 -cpu ? -name centos7 -drive file=/kvm/images\_creation/centos7.img,if=virtio,index=0,media=disk,cache=

writethrough,format=qcow2 -net nic,model=virtio,macaddr=52:54:00:11:22:33 -net tap,ifname=vif0.0,script=/kvm/scripts/net/qemu-natup,downscript=/kvm/scripts/net/qemu-natdown -vnc 0.0.0.0:0

\*\*\*\*\*这是在飞腾2000 ARM64架构下做的实验，这个实验成功的两个关键点，1、指定ARM64架构下的UEFI BIOS，2、vnc显示的界面是(qemu) 命令行界面，通过ctrl+alt+2进入操作系统启动界面。

[root@localhost ~]# qemu-kvm -m 8G -smp 4 -bios /usr/share/AAVMF/AAVMF\_CODE.fd -name kylin -cpu host -drive file=/var/lib/libvirt/images/ubuntu1804\_10G.qcow2,if=virtio,index=0,media=disk,cache=writethrough,format=qcow2 -net nic,model=virtio,macaddr=52:54:00:11:22:33 -net tap,ifname=vif0.0,script=/etc/qemu-natup,downscript=/etc/qemu-natdown -vnc :1

/etc/qemu-natup

#!/bin/bash

#

bridge="isbr"

net="10.0.0.0/8"

ifaddr="10.0.0.1/8"

checkbr() {

if brctl show | grep -i "^$1"; then

return 0

else

return 1

fi

}

initbr() {

brctl addbr $bridge

ip link set $bridge up

ip addr add $ifaddr dev $bridge

}

enable\_ip\_forward() {

sysctl -w net.ipv4.ip\_forward=1

}

setup\_nat() {

checkbr $bridge

if [ $? -eq 1 ]; then

initbr

enable\_ip\_forward

iptables -t nat -A POSTROUTING -s $net ! -d $net -j MASQUERADE

fi

}

if [ -n "$1" ]; then

setup\_nat

ip link set $1 up

brctl addif $bridge $1

exit 0

else

echo "Error: no interface specified."

exit 1

fi

/etc/qemu-natdown

#!/bin/bash

#

bridge="isbr"

remove\_rule() {

iptables -t nat -F

}

isalone\_bridge() {

if ! brctl show | awk "/^$bridge/{print \$4}" | grep "[^[:space:]]" &> /dev/null; then

ip link set $bridge down

brctl delbr $bridge

remove\_rule

fi

}

if [ -n "$1" ];then

ip link set $1 down

brctl delif $bridge $1

isalone\_bridge

exit 0

else

echo "Error: no interface specified."

exit 1

fi

libvirt工具栈：

支持的虚拟化技术：KVM, XEN, VMWARE, Qemu，LXC, OpenVZ;

libvirt中的术语：

node: 指物理节点

hypervisor：

domain: vm instances

安装：

# yum install libvirt libvirt-client python-virtinst virt-manager qemu-kvm

正常情况下qemu-kvm会被忽略，但是没有的话virt-manager会找不到hypervisor

CentOS 7：# yum install libvirt libvirt-client virt-install virt-manager qemu-kvm

libvirt和libvirtd的配置文件：

libvirt配置文件：/etc/libvirt/libvirt.conf

守护进程配置文件：/etc/libvirt/libvirtd.conf

域配置文件：xml格式

<vcpu placement='static'>2</vcpu>

<features>

</features>

<domain>

</domain>

Hypervisor的访问路径：

本地URL：

driver[+transport]:///[path][?extral-param]

driver: 驱动名称，例如qemu, xen, lxc

transport：传输方式

kvm使用qemu驱动，使用格式qemu:///system, 例如qemu:///system

远程URL：

driver[+transport]://[user@][host][:port]/[path][?extral-param]

例如：qemu://172.16.100.6/system

qemu+ssh://root@172.16.100.6/system

qemu+tcp://172.16.100.6/system

工具的使用：

(1) CLI: virt-install, virsh

(2) virt-manager

补充资料：virt-install使用文档

2.5.3.2 使用virt-install创建虚拟机并安装GuestOS

virt-install是一个命令行工具，它能够为KVM、Xen或其它支持libvirt API的hypervisor创建虚拟机并完成GuestOS安装；此外，它能够基于串行控制台、VNC或SDL支持文本或图形安装界面。安装过程可以使用本地的安装介质如CDROM，也可以通过网络方式如NFS、HTTP或FTP服务实现。对于通过网络安装的方式，virt-install可以自动加载必要的文件以启动安装过程而无须额外提供引导工具。当然，virt-install也支持PXE方式的安装过程，也能够直接使用现有的磁盘映像直接启动安装过程。

virt-install命令有许多选项，这些选项大体可分为下面几大类，同时对每类中的常用选项也做出简单说明。

◇ 一般选项：指定虚拟机的名称、内存大小、VCPU个数及特性等；

 -n NAME, --name=NAME：虚拟机名称，需全局惟一；

 -r MEMORY, --ram=MEMORY：虚拟机内在大小，单位为MB；

 --vcpus=VCPUS[,maxvcpus=MAX][,sockets=#][,cores=#][,threads=#]：VCPU个数及相关配置；

 --cpu=CPU：CPU模式及特性，如coreduo等；可以使用qemu-kvm -cpu ?来获取支持的CPU模式；

◇ 安装方法：指定安装方法、GuestOS类型等；

 -c CDROM, --cdrom=CDROM：光盘安装介质；

 -l LOCATION, --location=LOCATION：安装源URL，支持FTP、HTTP及NFS等，如http://172.16.0.1/cobbler/ks\_mirror/CentOS-7-x86\_64；

 --pxe：基于PXE完成安装；

 --livecd: 把光盘当作LiveCD；

 --os-type=DISTRO\_TYPE：操作系统类型，如linux、unix或windows等；

 --os-variant=DISTRO\_VARIANT：某类型操作系统的变体，如rhel5、fedora8等；

 -x EXTRA, --extra-args=EXTRA：根据--location指定的方式安装GuestOS时，用于传递给内核的额外选项，例如指定kickstart文件的位置，--extra-args "ks=http://172.16.0.1/class.cfg"

 --boot=BOOTOPTS：指定安装过程完成后的配置选项，如指定引导设备次序、使用指定的而非安装的kernel/initrd来引导系统启动等 ；例如：

 --boot cdrom,hd,network：指定引导次序；

 --boot kernel=KERNEL,initrd=INITRD,kernel\_args=”console=/dev/ttyS0”：指定启动系统的内核及initrd文件；

◇ 存储配置：指定存储类型、位置及属性等；

 --disk=DISKOPTS：指定存储设备及其属性；格式为--disk /some/storage/path,opt1=val1，opt2=val2等；常用的选项有：

 device：设备类型，如cdrom、disk或floppy等，默认为disk；

 bus：磁盘总线类型，其值可以为ide、scsi、usb、virtio或xen；

 perms：访问权限，如rw、ro或sh（共享的可读写），默认为rw；

 size：新建磁盘映像的大小，单位为GB；

 cache：缓存模型，其值有none、writethrouth（缓存读）及writeback（缓存读写）；

 format：磁盘映像格式，如raw、qcow2、vmdk等；

 sparse：磁盘映像使用稀疏格式，即不立即分配指定大小的空间；

--nodisks：不使用本地磁盘，在LiveCD模式中常用；

◇ 网络配置：指定网络接口的网络类型及接口属性如MAC地址、驱动模式等；

 -w NETWORK, --network=NETWORK,opt1=val1,opt2=val2：将虚拟机连入宿主机的网络中，其中NETWORK可以为：

 bridge=BRIDGE：连接至名为“BRIDEG”的桥设备；

 network=NAME：连接至名为“NAME”的网络；

其它常用的选项还有：

 model：GuestOS中看到的网络设备型号，如e1000、rtl8139或virtio等；

 mac：固定的MAC地址；省略此选项时将使用随机地址，但无论何种方式，对于KVM来说，其前三段必须为52:54:00；

--nonetworks：虚拟机不使用网络功能；

◇ 图形配置：定义虚拟机显示功能相关的配置，如VNC相关配置；

 --graphics TYPE,opt1=val1,opt2=val2：指定图形显示相关的配置，此选项不会配置任何显示硬件（如显卡），而是仅指定虚拟机启动后对其进行访问的接口；

 TYPE：指定显示类型，可以为vnc、sdl、spice或none等，默认为vnc；

 port：TYPE为vnc或spice时其监听的端口；

 listen：TYPE为vnc或spice时所监听的IP地址，默认为127.0.0.1，可以通过修改/etc/libvirt/qemu.conf定义新的默认值；

 password：TYPE为vnc或spice时，为远程访问监听的服务进指定认证密码；

 --noautoconsole：禁止自动连接至虚拟机的控制台；

◇ 设备选项：指定文本控制台、声音设备、串行接口、并行接口、显示接口等；

 --serial=CHAROPTS：附加一个串行设备至当前虚拟机，根据设备类型的不同，可以使用不同的选项，格式为“--serial type,opt1=val1,opt2=val2,...”，例如：

 --serial pty：创建伪终端；

 --serial dev,path=HOSTPATH：附加主机设备至此虚拟机；

 --video=VIDEO：指定显卡设备模型，可用取值为cirrus、vga、qxl或vmvga；

◇ 虚拟化平台：虚拟化模型（hvm或paravirt）、模拟的CPU平台类型、模拟的主机类型、hypervisor类型（如kvm、xen或qemu等）以及当前虚拟机的UUID等；

 -v, --hvm：当物理机同时支持完全虚拟化和半虚拟化时，指定使用完全虚拟化；

 -p, --paravirt：指定使用半虚拟化；

 --virt-type：使用的hypervisor，如kvm、qemu、xen等；所有可用值可以使用’virsh capabilities’命令获取；

◇ 其它：

 --autostart：指定虚拟机是否在物理启动后自动启动；

 --print-xml：如果虚拟机不需要安装过程(--import、--boot)，则显示生成的XML而不是创建此虚拟机；默认情况下，此选项仍会创建磁盘映像；

 --force：禁止命令进入交互式模式，如果有需要回答yes或no选项，则自动回答为yes；

 --dry-run：执行创建虚拟机的整个过程，但不真正创建虚拟机、改变主机上的设备配置信息及将其创建的需求通知给libvirt；

 -d, --debug：显示debug信息；

尽管virt-install命令有着类似上述的众多选项，但实际使用中，其必须提供的选项仅包括--name、--ram、--disk（也可是--nodisks）及安装过程相关的选项。此外，有时还需要使用括--connect=CONNCT选项来指定连接至一个非默认的hypervisor。

使用示例：

(1) # virt-install -n "centos6" -r 512 --vcpus=2 -l http://172.16.0.1/cobbler/ks\_mirror/CentOS-6.6-x86\_64/ -x "ks=http://172.16.0.1/centos6.x86\_64.cfg" --disk path=/images/kvm/centos6.img,size=120,sparse --force -w bridge=br100,model=virtio

(2) 下面这个示例创建一个名为rhel5的虚拟机，其hypervisor为KVM，内存大小为512MB，磁盘为8G的映像文件/var/lib/libvirt/images/rhel5.8.img，通过boot.iso光盘镜像来引导启动安装过程。

# virt-install \

--connect qemu:///system \

--virt-type kvm \

--name rhel5 \

--ram 512 \

--disk path=/var/lib/libvirt/images/rhel5.img,size=8 \

--graphics vnc \

--cdrom /tmp/boot.iso \

--os-variant rhel5

(3) 下面的示例将创建一个名为rhel6的虚拟机，其有两个虚拟CPU，安装方法为FTP，并指定了ks文件的位置，磁盘映像文件为稀疏格式，连接至物理主机上的名为brnet0的桥接网络：

# virt-install \

--connect qemu:///system \

--virt-type kvm \

--name rhel6 \

--ram 1024 \

--vcpus 2 \

--network bridge=brnet0 \

--disk path=/VMs/images/rhel6.img,size=120,sparse \

--location ftp://172.16.0.1/rhel6/dvd \

--extra\_args “ks=http://172.16.0.1/rhel6.cfg” \

--os-variant rhel6 \

--force

(4) 下面的示例将创建一个名为rhel5.8的虚拟机，磁盘映像文件为稀疏模式的格式为qcow2且总线类型为virtio，安装过程不启动图形界面（--nographics），但会启动一个串行终端将安装过程以字符形式显示在当前文本模式下，虚拟机显卡类型为cirrus：

# virt-install \

--connect qemu:///system \

--virt-type kvm \

--name rhel5.8 \

--vcpus 2,maxvcpus=4 \

--ram 512 \

--disk path=/VMs/images/rhel5.8.img,size=120,format=qcow2,bus=virtio,sparse \

--network bridge=brnet0,model=virtio

--nographics \

--location ftp://172.16.0.1/pub \

--extra-args "ks=http://172.16.0.1/class.cfg console=ttyS0 serial" \

--os-variant rhel5 \

--force \

--video=cirrus

(5) 下面的示例则利用已经存在的磁盘映像文件（已经有安装好的系统）创建一个名为rhel5.8的虚拟机：

# virt-install \

--name rhel5.8

--ram 512

--disk /VMs/rhel5.8.img

--import

virt-install -n cirros -r 128 --vcpus=1,maxvcpus=4 --disk /images/cirros/cirros-no\_cloud-0.3.0-i386-disk.img --network bridge=br0,model=virtio --import --serial=pty --console=pty --nographics

^]

亲测试验：

[root@localhost qemu]# virt-install --name pxe\_centos7 --memory 2048 --vcpus 2 -l http://192.168.51.236/ -x "ks=http://192.168.51.236/ks.cfg" --disk path=/images/pxe\_centos7,size=60,sparse=yes --network bridge=virbr0 --graphics vnc --hvm

\*\*\*\*\*安装时出现问题dracut#错误(no space left on device)，起初以为是硬盘分区的问题，后来反复偿试，这里的no space指的是内存空间不足，之间分配的是1G ,改成2G 就好了，所以总结一下就是错误信息一定要把握准确，安装初期都是在内存中完成，no space指内存不足就很合情理

注意：每个虚拟机创建后，其配置信息保存在/etc/libvirt/qemu目录中，文件名与虚拟机相同，格式为XML，以后可以用virsh create来直接创建

virsh的几个常用命令：

virt-install：创建虚拟机，并安装OS；也可创建虚拟机并导入Image文件；

根据xml文件创建：

create：创建并启动

define：创建但不启动

关闭domain：

destroy

shutdown

reboot

删除domain：

undefine

连接至console：

console

列出：

list

附加或拆除disk：

attach-disk

detach-disk

附加或拆除网卡：

attach-interface

detach-interface

保存状态至磁盘文件或从磁盘文件恢复：

save

restore

暂停于内存或继续运行：

suspend

resume

virt-manager：GUI工具

http://www.linux-kvm.org/page/Management\_Tools

补充资料：Qemu监视器

图形窗口：ctrl+alt+2, ctrl+alt+1

文本（-nographic）: Ctrl+a, c

算定义minitor接口的输出位置：-monitor /dev/XXX

-monitor stdio

常用命令：

help: 显示帮助

help info

info: 显示系统状态

info cpus

info tlb

commit：

change：

change vnc password

device\_add和device\_del:

usb\_add和usb\_del

savevm, loadvm, delvm

创建、装载及删除虚拟机快照；

migrate, migrate\_cancel

cpu CPU\_index

log和logfile：

sendkey

system\_powerdown, system\_reset, system\_wakeup

q或quit: 退出qemu模拟器，qemu进程会终止

虚拟化管理工具：

http://www.linux-kvm.org/page/Management\_Tools

网络虚拟化技术

OpenVSwitch: 虚拟交换机

VLAN, VXLAN

虚拟路由器

什么是VLAN？

Virtual LAN; LAN即为广播帧能够到的节点范围，也即能够直接通信的范围；

VLAN:

基于MAC地址

基于交换机Port实现

基于IP地址实现

基于用户实现

交换机接口的类型：

访问链接：access link

汇聚链接：trunc link

VLAN的汇聚方式：

IEEE 802.1q

ISL: Inter Switch Link

VLAN间路由：

路由器：

访问链接：router为每个VLAN提供一个接口

汇聚链接：router只向交换提供一个接口

三层交换机

虚拟化技术：

cpu, memory, i/o

IaaS：Infrastructure as a Service

PaaS：Platform as a Service

Linux内核：

namespace

文件系统隔离；

网络隔离： 主要用于实现网络资源的隔离，包括网络设备、IPv4以及IPv6地址、IP路由表、防火墙、/proc/net、/sys/class/net以及套接字等；

IPC隔离；一个名称空间内的各进程通信，不能夸名称空间

用户和用户组隔离：

PID隔离：对名称空间内的PID重新标号，两个不同的名称空间可以使用相同的PID；

UTS隔离：Unix Time-sharing System，提供主机名称和域名的隔离；

cgroups

用于完成资源配置；用于实现限制被各namespace隔离起来的资源，还可以为资源设置权重 、计算使用量、完成各种所需的管理任务等；

Linux Network NameSpace：

注意：netns在内核实现，其控制功能由iproute所提供的netns这个OBJECT来提供；CentOS6.6提供的iproute不具有此OBJECT，需要依赖于OpenStack Icehouse的EPEL源来提供；

1、使用netns

ip netns list

ip netns add NAME

ip netns del NAME

ip netns exec NAME COMMAND

2、使用虚拟以太网卡

ip link add FRONTEND-NAME type veth peer name BACKEND-NAME

网络虚拟化：

复杂的虚拟化网络：

netns

OpenVSwitch

OVS：基于C语言研发；特性：

802.1q, trunk, access

NIC bonding

NetFlow, sFlow

QoS配置及策略

GRE, VxLAN,

OpenFlow

OVS的组成部分：

ovs-vswitchd: OVS daemon, 实现数据报文交换功能，和Linux内核兼容模块一同实现了基于流的交换技术；

ovsdb-server：轻量级的数据库服务，主要保存了整个OVS的配置信息，例如接口、交换和VLAN等等；ovs-vswithed的交换功能基于此库实现；

ovs-dpctl: 配置交换机内核模块，控制转发规则

ovs-vsctl：用于获取或更改ovs-vswitchd的配置信息，其修改操作会保存至ovsdb-server中；

ovs-appctl

ovsdbmonitor

ovs-controller

ovs-ofctl

ovs-pki

ovs-vsctl命令的使用：

show: ovsdb配置内容查看

add-br NAME：添加桥设备；

list-br: 显示所有已定义BRIDGE

del-br BRIDGE: 删除桥

add-port BRIDGE PORT: 将PORT添加至指定的BRIDGE

list-ports BRIDGE: 显示指定BRIDGE上已经添加的所有PORT

del-port [BRIDGE] PORT: 从指定BRIDGE移除指定的PORT

/etc/if-up脚本：

#!/bin/bash

#

bridge=br-in

if [ -n "$1" ]; then

ip link set $1 up

sleep 1

ovs-vsctl add-port $bridge $1

[ $? -eq 0 ] && exit 0 || exit 1

else

echo "Error: no port specified."

exit 2

fi

/etc/if-down脚本：

#!/bin/bash

#

bridge=br-in

if [ -n "$1" ]; then

ip link set $1 down

sleep 1

ovs-vsctl del-port $bridge $1

[ $? -eq 0 ] && exit 0 || exit 1

else

echo "Error: no port specified."

exit 2

fi

GRE: Generic Routing Encapsulation，通用路由封装；是一种隧道技术；

OpenStack

IaaS云栈，CloudOS;

私有云

公有云

混合云

IaaS (OpenStack, CloudStack)、PaaS(Docker, Openshift)、SaaS

DBaaS、FWaaS

按需提供VM

Openstack的组件：

Compute: 代码名Nova，管理VM的整个生命周期，主要职责包括启动、调度VMs;

Networking：代码名Neutron(早期叫Quantum，独立之前为nova-netwroking)；为Openstack提供NCaaS的功能；插件化设计，支持众多流行的网络管理插件；

Object Storage: 代码名swift；分布式存储，基于RESTful的API实现非结构化数据对象的存储及检索；

Block Storage：代码名为Cinder（早期由Nova提供，代码为nova-storate），为VMs提供持久的块存储能力；

Identity: 代码为Keystone；为Openstack中的所有服务提供了认证、授权以及端点编录目录；

Image: 代码名Glance，用于存储和检索磁盘映像文件；

Dashboard: 代码名为Horizon，WebGUI;

Telemetry: 代码名为Ceilometer，用于实现监控和计量服务的实现；

Orachestration: 代码名为Heat，用于多组件联动；

Database：代码为Trove，提供DBaaS服务的实现；

Data processing：代码为sahara，用于在OpenStack中实现Hadoop的管理；

OpenStack安装配置：

Identity: 主要有两个功能

用户管理：认证和授权

认证方式有两种：

token

账号和密码

服务目录：所有可用服务的信息库，包含其API endpoint路径

几个核心术语：

User, Role, Tanent, Service, Endpoint, Token

epel6 icehouse 安装源：https://repos.fedorapeople.org/repos/openstack/openstack-icehouse/

管理工具：keystone-manager

客户端程序：keystone

Image Service:

代码名：Glance，用于在OpenStack中注册、发现及获取VM映像文件；

VM的映像文件存储于何处？

普通文件系统、对象存储系统（swift）、S3存储，以及HTTP服务上；

磁盘映像文件：

(1) 制作

Oz(KVM)

VMBuilder(KVM, XEN)

VeeWee(KVM)

imagefactory

(2) 获取别人制作模板

CirrOS

Ubuntu

Fedora

OpenSUSE

Rackspace云映像文件生成器

OpenStack中的磁盘映像文件要满足以下要求：

(1) 支持由Openstack获取其元数据信息；

(2) 支持对映像文件的大小进行调整；

推荐书籍：《奇点临近》，《乌合之众》

回顾：

Keystone: Identity Service

认证、授权和服务编录

认证方式：

token

credential: 账号密码

Glance：Image Service

存储、查询及获取

glance-registry

storage adapter：

FileSystem: /var/lib/glance/images

Swift

OpenStack：

Compute Service

代码为Nova

Supporting Service:

AMQP: Advanced Messaging Queue Protocol

Apache Qpid(5672/tcp), RabbitMQ, ZeroMQ

Database

组件：

API:

nova-api, nova-api-metadata

Compute Core:

nova-compute, nova-scheduler, nova-conductor

Network for VMs:

nova-network, nova-dhcpagent

Console Interface:

nova-consoleauth, nova-novncproxy, nova-xvpnvncporxy, nova-cert

Command line and other interfaces:

nova, nova-manage

Compute服务的角色：

管理角色

hypervisor

安装qpid:

# yum install qpid-cpp-server

编辑配置文件，设置"auth=no"

# service qpidd start

注意：配置compute节点时，额外需要在[DEFAULT]配置段设定的参数：

vif\_plugging\_timeout=10

vif\_plugging\_is\_fatal=false

Network Service:

代码为：Neutron，早期叫Quantum

有两种配置机制：

legacy network

Neutron

Neutron Server：controller

Network Node:

Compute Nodes: Computes

功能：

OVS, l3(netns), dhcpagent, NAT, LBaaS, FWaaS, IPSec VPN

Networking API

network, subnet, port

Network: 隔离的2层网络，类似于VLAN；

Subnet: 有着关联配置状态的3层网络，或者说是由Ipv4或ipv6定义的地址块形成的网络；

Port: 将主机连入网络设备的连接接口；

插件：

plug-in agent: netutron-\*-agent

dhcp agent

l3 agent

l2 agent

OpenStack中物理网络连接架构：

管理网络(management network)：

数据网络(data network):

外部网络(external network):

API网络

Tenant network: tenant内部使用的网络；

Flat: 所有VMs在同一个网络中，不支持VLAN及其它网络隔离机制；

Local: 所有的VMs位于本地Compute节点，且与external网络隔离；

VLAN：通过使用VLAN的IDs创建多个providers或tenant网络；

VxLAN和GRE：

provider network: 不专属于某tenant，为各tenant提供通信承载的网络；

DashBoard:

Python Django

Web Framework

注意事项：

1、Neutron的配置文件中要把auth\_uri换成identity\_uri;

2、各配置文件属组应该为相应的服务的运行者用户身份，否则其将无法访问导致服务启动失败；

Block Storage Service

代码名：Cinder

组件：

cinder-api

cinder-volume

cinder-scheduler

部署工具：

fuel: mirantis

devstack

测试玩玩：

ip netns add r1

ip netns add r2

brctl add br\_ex

ip link set br\_ex up

ip addr del xxx dev eno16777728; ip addr add xxx dev br\_ex; brctl addif br\_ex eno16777728

brctl addr br\_in

ip link set br\_in up

ip link add veth1.1 type veth peer name veth1.2

ip link show

ip link set veth1.1 netns r1

ip link set veth1.2 netns r2

ip link show

正经测试：

创建两个虚拟机实例

虚拟机B

虚拟机A

路由名称空间

网桥B

网桥A

先打开核心转发，免得出错

#!/bin/bash

#

bridge=br\_in

if [ -n “$1” ]; then

　　　ip link set $1 up

　　　brctl addif $bridge $1

　　　[ $? -eq 0 ] && exit 0 || exit1

else

　　　echo “Error: no interface specified.”

　　　exit 1

fi

ip netns add r1

ip netns add r2

ip link add r\_in\_r1.1 type veth peer name r\_in\_s1.2

ip link show

ip link set r\_in\_r up

ip link set r\_in\_s up

一半在桥上

brctl addif br\_in r\_in\_s

brctl show

一半在router命称空间上

ip link set r\_in\_r netns r1

ip link show

ip netns exec r1 ifconfig -a

ip netns exec r1 ip link set r\_in\_r name eth0 改个名

ip netns exec r1 ip link set eth0 up

ip netns exec r1 ifconfig eth0 xxx up

可以设置虚拟机的IP地址，添加route

ip link add rexr type veth peer name rexs

ip link show

brctl addif br\_ex rexs

ip link set rexs up

brctl show

ip link set rexr netns r1

ip netns exec r1 ip link set rexr name eth1

ip netns exec r1 ifconfig eth1 xxx up

ip netns exec r1 ping ...... 可以ping出去，但是回不来

ip netns exec r1 iptables -t nat -A POSTROUTING -s 10.0.1.0/24 -! -d 10.0.1.0/24 -j SNAT --to-source 172.16.100.78

ip netns exec r1 dnsmasq --dhcp-range 10.0.1.100,10.0.1.200

虑拟机可以获得地址试

# udhcpc