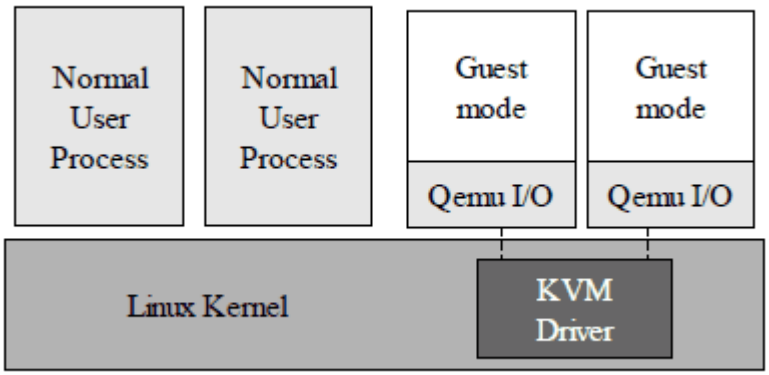
什么是KVM

kvm是一个完整的全虚拟化解决方案. 当前主机操作系统支持linux系统和x
86架构. 它包括了一个可加载的内核模块(kvm.ko)和一个用户空间组件. 通过使用kvm
虚拟化技术, 你可以运行多个未修改的linux或windows
虚拟机. 每一个虚拟机有自己的虚拟硬件: 一个网卡, 磁盘, 图形适配器, 等.

KVM支持全虚拟化和半虚拟化. 你可以通过给当前内核打kvm-paravirt-patches补丁来支持半虚拟化.



参考资料

> - http://virt.kernelnewbies.org/KVM
> - Redhat Virtualization Document Index
> - OSSLab Virtualization Documents

KVM包含两大组件

> 1. a device driver for managing the virtualization hardware; this driver exposes its capabilities via a character device /dev/kvm
> 2. a user-space component for emulating PC hardware; this is a lightly modified QEMU process
>
> QEMU is a well known processor emulator written by French computer wizard Fabrice Bellard.

KVM的优缺点

kvm具有以下**优点**

> 1. 速度 kvm的cpu效率和磁盘效率绝对是最优秀的。
> 2. 稳定 目前的感受是最稳定的，不比 vmware 差。
> 3. 扩展性 kvm是一个新的项目，以后回逐步在图形效率，3d和硬件上面得到linux最基础的支持。

当然也有**缺点**

> 1. 使用不太方便。很多人惧怕命令行的东西，但是稍微注意一下，会发现命令行是那么的清晰和简单。
> 2. 目前有些功能不太完善，比如图形性能，usb等。
> 3. cpu需要支持VM技术，否则不能体现 kvm 的优势。但这种基于硬件的技术也正是性能的保证。

Ubuntu+KVM+Kqemu实战

系统背景

```
Ubuntu 9.04
kvm-84
```

先了解qemu和kvm的关系

> 1. kvm基本是一个独立的系统，只不过借用了qemu
>   的代码和操作界面。也就是说，你在命令行中如果使用 qemu 的地方，只需要题换成 kvm
>   ，其他的参数基本都不用修改就可以运行，这样的效率是 qemu 的很多倍。
> 2. 如果要用kvm，你的cpu必须支持 vm 技术，否则和 qemu 的效率基本相同。
>
> 如果你对KVM指令很困惑的话，可以先安装qemu GUI界面进行配置，然后导出qemu命令行就可以了解KVM的用法

如何判断cpu是否支持 vm 技术

首先使用

```
<span class="kw2">egrep</span> <span class="st0">'(vmx|svm)'</span> <span
class="sy0">/</span>proc<span class="sy0">/</span>cpuinfo
```

如果有输出，说明你的cpu支持VM

注意，大多BIOS缺省是关掉了vm支持的，你需要首先设置你的bios。

CPU支持硬件VM虚拟技术（BIOS里设置），就能够加载**kvm-intel or kvm-amd**模块

安装软件

```
sudo apt-get install kqemu-common kqemu-source kvm
```

如果想使用**kqemu**内核加速模块，kqemu-source
软件包必须安装，它会根据内核版本编译相应模块提供安装。

检查模块

```
geminis<span class="sy0">@</span>geminis-ubuntu:~$ <span
class="kw2">lsmod</span> <span class="sy0">|</span>less
Module                  Size  Used by

kqemu                  <span class="nu0">137892</span>  <span
class="nu0">0</span>
bridge                  <span class="nu0">56340</span>  <span
class="nu0">0</span>
kvm_intel               <span class="nu0">50624</span>  <span
class="nu0">0</span>
kvm                    <span class="nu0">152640</span>  <span
class="nu0">1</span> kvm_intel
```

创建虚拟硬盘

kvm-img create -f 参数

```
-f 参数表示创建文件格式，后边所带的子参数qcow/qcow2 表示KVM默认镜像格式，vmdk表示VMWARE的磁盘格式。
```

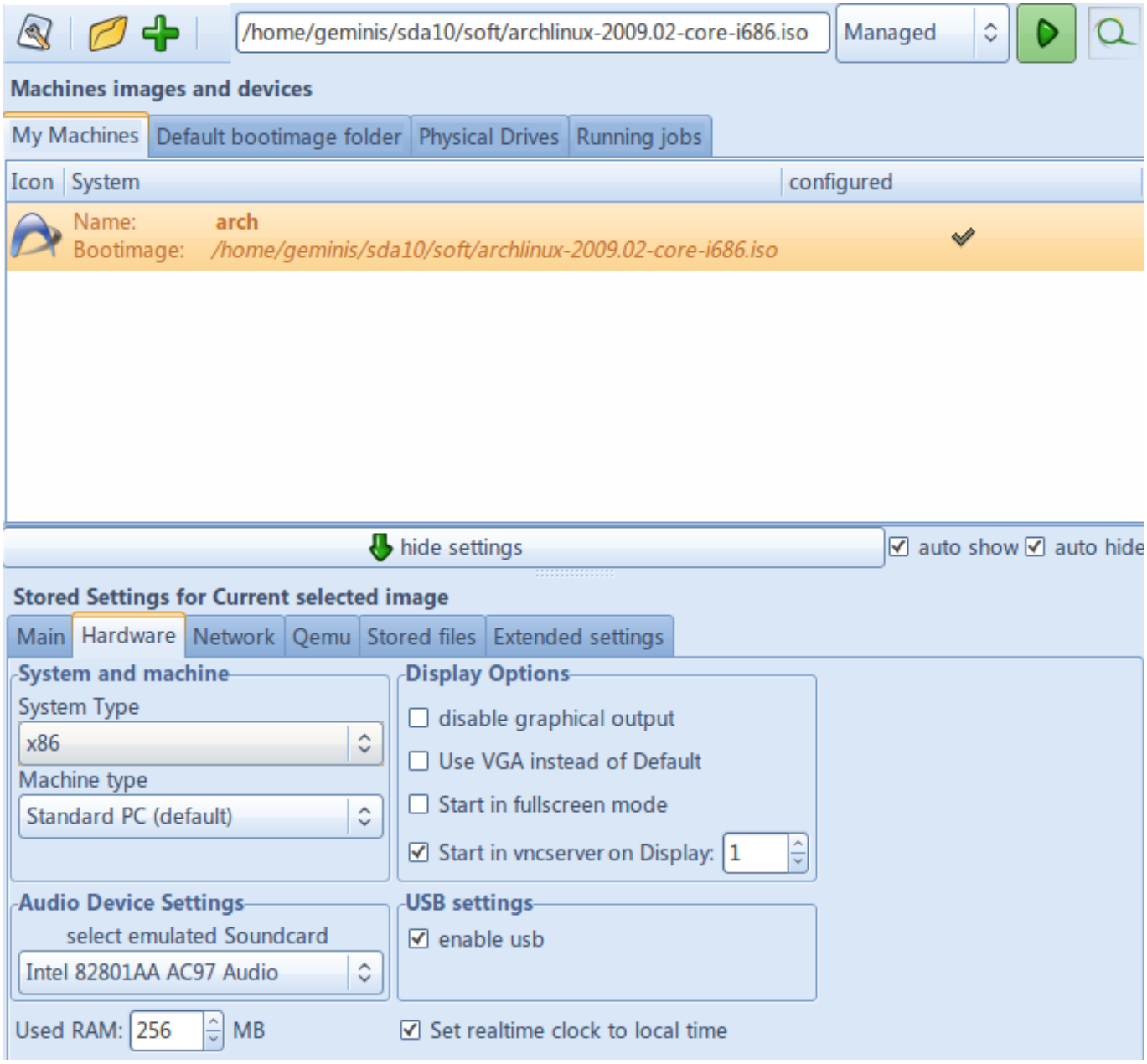
建议大家不要创建qcow格式的磁盘文件.它生成的磁盘文件比较占用空间.该命令是常规命令.可以使用vmdk格式，这两种创建方式在使用的过程中没有很大区别,唯一的区别在于磁盘占用率上。

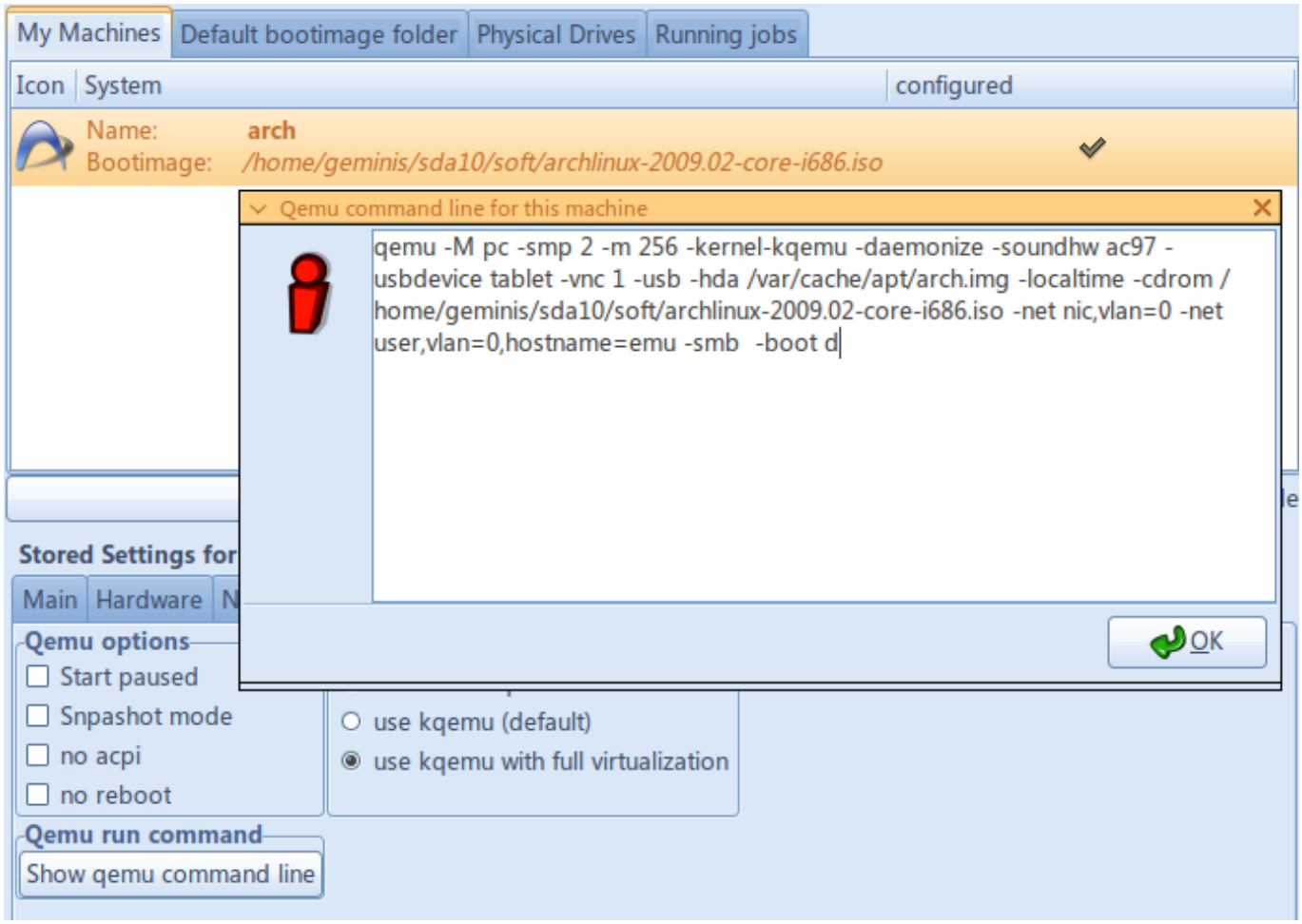
```
<span class="co0"># sudo kvm-img create -f vmdk arch.img 2G</span>

Formatting <span class="st0">'arch.img'</span>, <span
class="re2">fmt=</span>vmdk, <span class="re2">size=</span><span
class="nu0">2097152</span> kB

<span class="co0"># chmod 777 arch.img #允许虚拟机能够读写镜像文件</span>
```

启动qemulator配置Archlinux





KVM 网络配置

参考资料

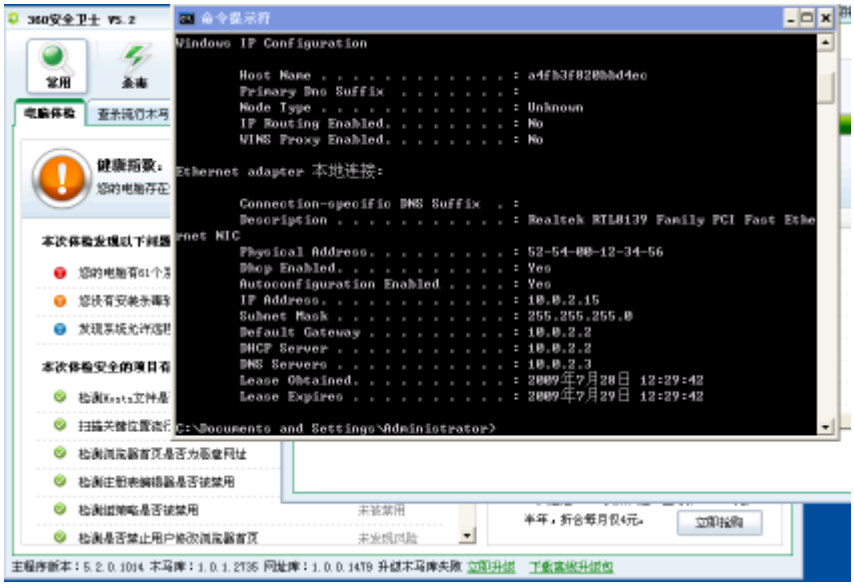
- OSSLab Network Setting
- virt-manager bridged networking

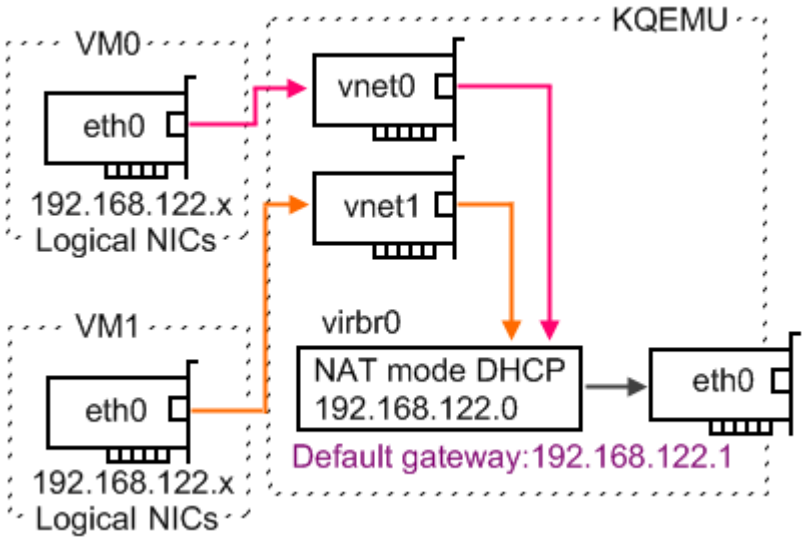
KVM支持三种网络模式，跟VMware所支持的一样：NAT, Host-Only, Bridge

DHCP+NAT 模式

这是最简单的一种，直接在启动参数中加上-net user即可，主机打开DHCP功能，客户机通过DHCP获取IP，该置下客户机可以上网，但不能跟主机通讯。

```
kvm  -net nic -net user
```





Host-Only

使用tap/tun虚拟出一个网卡，需要内核支持tap/tun模块（一般都支持），还需要配置iptables转发，因为tap网卡上的ip不能和主机上ip
在同一个网段。客户使用该网卡可以跟主机通讯也可以上网，但局域网内的其他机器不能直接访问客户机，需要主机通过端口转发或者其他方式来访问。

我的简单配置（内核需要支持tap/tun）：

（1）修改/etc/sysctrl.conf, 将net.ipv4.ip_forward=1打开，也就是ip转发功能打开，修改完后需要reboot机器。

（2）建立一个脚本**/etc/kvm-ifup.sh**,内容很简单：

```
<span class="co0">#!/bin/bash</span>
<span class="kw2">sudo</span> ifconfig $<span class="nu0">1</span> <span
class="nu0">172.0</span><span class="nu0">.100</span><span
class="nu0">.1</span> netmask <span class="nu0">255.255</span><span
```

```
class="nu0">.255</span><span class="nu0">.0</span> up
```

该脚本就是配置tap虚拟网卡设备的ip。\$1通常会是tap0，根据你在kvm启动参数中给定的名字而定。

（3）在qemu
启动参数中给定相关的虚拟网卡设备名，和相关的设备初始化脚本，也就是上面的这个脚本：

```
kvm  -hda <span class="sy0">/</span>virtualOS<span
class="sy0">/</span>winxp.img -m <span class="nu0">512</span> -no-frame
-localtime -no-acpi      \
         -net nic,<span class="re2">model=</span>virtio -net tap,<span
class="re2">ifname=</span>tap0,<span class="re2">script=</span><span
class="sy0">/</span>etc<span class="sy0">/</span>kvm-ifup.<span
class="kw2">sh</span>   \
         -usb -usbdevice tablet
```

（4）在客户机中，手动的配置ip，网关和DNS，跟tap虚拟网卡的ip同一网段，而网关就是虚拟网卡的ip，比如我的xp配置：

```

IP: 172.0.100.101
MASK: 255.255.255.0
GATEWAY:172.0.100.1
```

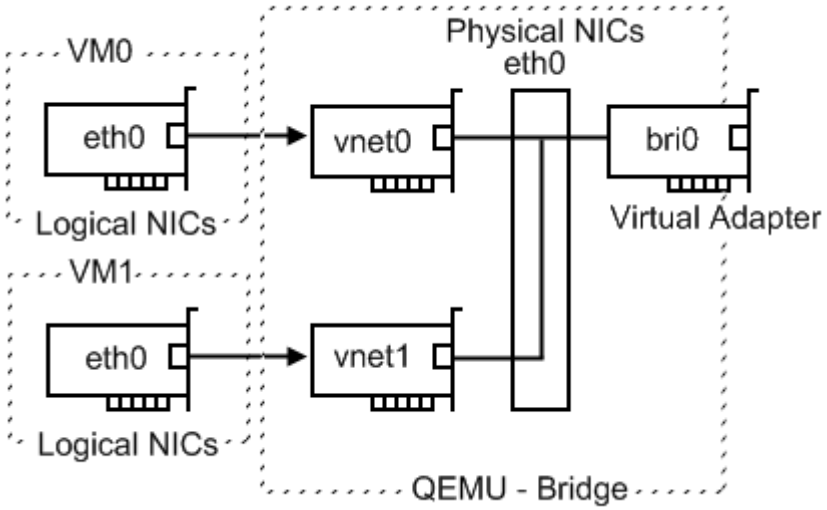
（5)配置iptables的NAT:

```
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

我是放在一个启动脚本里，每次启动kvm会执行一下，也可以在主机系统启动的时候就添加这条规则。

Bridge 模式

这种应该是功能最全的一种，虚拟客户机的ip
和主机处在同一个网段，客户机就如局域网中的一台主机一样，既可以和主机通讯也可以上网，同时网路中的其他主机也可以访问客户机而不需要主机转发。



KVM仅支持 **tap 或 tun** 网卡设备，因此，我们需要安装创建TAP界面的工具

```
<span class="kw2">sudo</span> apt-get <span class="kw2">install</span>
bridge-utils
<span class="kw2">sudo</span> apt-get <span class="kw2">install</span>
uml-utilities
```

```
kvm -net nic -net tap,<span class="re2">ifname=</span>tap0,<span
class="re2">script=</span><span class="sy0">/</span>etc<span
class="sy0">/</span>kvm<span class="sy0">/</span>kvm-ifup <span
class="re2">downscript=</span><span class="sy0">/</span>etc<span
class="sy0">/</span>kvm<span class="sy0">/</span>kvm-ifdown ,<span
class="re2">hostname=</span>kvm-host
```

/etc/kvm/kvm-ifup脚本

```
<span class="co0">#!/bin/sh</span>
<span class="re2">BRIDGE=</span>br0
<span class="re2">OLD_IF=</span>eth0

<span class="co0">## 获取原来主系统 eth0 的 IP 和网关</span>
<span class="re2">IP=</span>`ifconfig <span class="re1">$OLD_IF</span><span
class="sy0">|</span>grep <span class="st0">"inet addr"</span><span
class="sy0">|</span>awk <span class="st0">'{print $2}'</span><span
class="sy0">|</span>sed <span class="st0">'s/addr://'</span>`
<span class="re2">GATEWAY=</span>`route -n<span class="sy0">|</span>grep
^<span class="nu0">0.0</span><span class="nu0">.0</span><span
class="nu0">.0</span><span class="sy0">|</span>awk <span class="st0">'{print
$2}'</span>`

<span class="co0">## 删除 eth0 的 IP</span>
ip addr del <span class="re1">$IP</span> dev <span class="re1">$OLD_IF</span>

<span class="co0">## 激活 tap0 ($1 是qemu传递过来的网卡名称，即tap0)</span>
```

```
ip <span class="kw2">link</span> <span class="kw1">set</span> $<span
class="nu0">1</span> up

<span class="co0">##停止并删除存在的网桥</span>
ip <span class="kw2">link</span> <span class="kw1">set</span> <span
class="re1">$BRIDGE</span> down <span class="sy0">&>/</span>dev<span
class="sy0">/</span>null
brctl delbr <span class="re1">$BRIDGE</span> <span
class="sy0">&>/</span>dev<span class="sy0">/</span>null

<span class="co0">##添加网桥接口 br0</span>
brctl addbr <span class="re1">$BRIDGE</span>

<span class="co0">##将 eth0加入网桥</span>
brctl addif <span class="re1">$BRIDGE</span> <span class="re1">$OLD_IF</span>

<span class="co0">##将 tap0加入网桥</span>
brctl addif <span class="re1">$BRIDGE</span> $<span class="nu0">1</span>

<span class="co0">##启动网桥</span>
ifconfig <span class="re1">$BRIDGE</span> <span class="re1">$IP</span> up

<span class="co0">##添加默认路由</span>
route add default gw <span class="re1">$GATEWAY</span>
```

/etc/kvm/kvm-ifdown脚本

```
<span class="co0">#!/bin/sh</span>
<span class="re2">BRIDGE=</span>br0
<span class="re2">OLD_IF=</span>eth0

<span class="re2">IP=</span>`ifconfig <span class="re1">$BRIDGE</span><span
class="sy0">|</span>grep <span class="st0">"inet addr"</span><span
class="sy0">|</span>awk <span class="st0">'{print $2}'</span><span
class="sy0">|</span>sed <span class="st0">'s/addr://'</span>`
<span class="re2">GATEWAY=</span>`route -n<span class="sy0">|</span>grep
^<span class="nu0">0.0</span><span class="nu0">.0</span><span
class="nu0">.0</span><span class="sy0">|</span>awk <span class="st0">'{print
$2}'</span>`

<span class="co0">##停止并删除存在的网桥</span>
ip addr del <span class="re1">$IP</span> dev <span class="re1">$BRIDGE</span>
ip <span class="kw2">link</span> <span class="kw1">set</span> <span
class="re1">$BRIDGE</span> down
brctl delbr <span class="re1">$BRIDGE</span>

<span class="co0">##恢复原来的网络配置</span>
ifconfig <span class="re1">$OLD_IF</span> <span class="re1">$IP</span> up
```

```
route add default gw <span class="re1">$GATEWAY</span>
```

不过如果在无线网卡上做桥接会遇到一定的问题，很可能就不成功，因为KVM
跟无线网卡驱动有一定的关系

参考 http://www.linuxfoundation.org/en/Net:Bridge

KVM 基本操作

KVM 的命令格式

```
kvm [options] [disk_image]
```

常用参数的说明

```
-hda/-hdb file 使用'file'作为 IDE 硬盘 0/1 的虚拟磁盘。
-hdc/-hdd file 使用'file'作为 IDE 硬盘 2/3 的虚拟磁盘。
-cdrom file 使用'file'作为 IDE CD-ROM光盘镜像（文件应该是ISO类型）(不可以跟 -hdc
参数同时使用)。
-cdrom /dev/hdX 使用实体光驱机作为IDE CD-ROM，X为实际上光驱机从系统取得的代号（a,b,c
or d）
-boot [a|d|c] 使用软盘a，光盘d，或者硬盘c启动.
-m memorysize 设定虚拟机器使用多少内存(预设为 128MB)。
-soundhw c1,...    开启音效支援，使用 -soundhw ?列出可用的声卡，若要开启全部，请使用 -
soundhw all。
-usb 允许使用usb设备.
-usbdevice 名字    添加一个usb设备“名字”.
-net nic 创建一块新的网卡.
-kernel-kqemu 开启 KQEMU 完全加速模式，预设则为 User Mode。
                                        (AMD64 用户如果是自行编译 QEMU，必须使用
qemu-system-x86_64 才能带此参数)
```

预设会有网路(User Mode)，并由 DHCP 自动发配 IP。其余参数及详细说明请自行参阅 KVM 的 Manual Page。

KVM 操作范例

```
<span class="co0">#-kernel-kqemu </span>

kvm -hda <span class="sy0">/</span>var<span class="sy0">/</span>cache<span
class="sy0">/</span>apt<span class="sy0">/</span><span
class="kw2">arch</span>.img   \
```

```
        -cdrom <span class="sy0">/</span>home<span
class="sy0">/</span>geminis<span class="sy0">/</span>sda10<span
class="sy0">/</span>soft<span class="sy0">/</span>archlinux<span
class="nu0">-2009.02</span>-core-i686.iso   \
        -M pc -smp <span class="nu0">2</span> -m <span class="nu0">256</span>
\
        -net nic,<span class="re2">vlan=</span><span class="nu0">0</span>
-net tap,<span class="re2">vlan=</span><span class="nu0">0</span>,<span
class="re2">ifname=</span>tap0,<span class="re2">script=</span><span
class="sy0">/</span>etc<span class="sy0">/</span>kvm<span
class="sy0">/</span>kvm-ifup,<span class="re2">downscript=</span><span
class="sy0">/</span>etc<span class="sy0">/</span>kvm<span
class="sy0">/</span>kvm-ifdown,<span class="re2">hostname=</span>kvm-winxp \
        -soundhw ac97   \
        -localtime   \
        -clock rtc    \
        -usb            \
        -smb           \
        -usbdevice tablet -vnc <span class="nu0">127.0</span><span
class="nu0">.0</span><span class="nu0">.1</span>:<span class="nu0">0</span>
-redir tcp:<span class="nu0">3389</span>::<span class="nu0">3389</span>  \
        -boot d \
        -daemonize
```

此行命令表示：

- hda 设定为 'arch.img'
- CD-ROM 设定为 'archlinux-2009.02-core-i686.iso'
- 机器类型为x86标准PC，2个CPU，内存为 256MB
- 网络为用户预设模式
- 开启 Sound Card 'ac97'
- 使用本地时间（ **一定要加这个参数，不然虚拟机时间会有问题** ）
- 使用rtc时钟（ **开启此项，有可能会导致window无法启动，酌情关闭** ）
- 开启USB支持
- 使用SMB文件共享
- 打开远程桌面支持，并把虚拟机的远程桌面服务器端口映射到宿主机的端口，等同于windows 的远程桌面
- 从光驱开机
- 开启 KQEMU 完全加速模式（ **如果是KVM直接运行，则不需要再加速** ）
- 后台运行(可以使用远程桌面控制，详见下面)

当虚拟机在后台运行时，你有两种方式查看虚拟机的界面

1. 使用远程桌面工具，地址 localhost , 端口 5900。
2. 使用远程终端，地址 localhost 端口 3389

如果你的虚拟机还没有配置远程桌面，请先用第一种方式访问虚拟机界面，然后设定虚拟机（我的是XP

）远程桌面方式，然后就可以了。

下一步是整合虚拟机到你的桌面，就和我的贴图一样，使用以下命令

```
rdesktop localhost:<span class="nu0">3389</span> -u danny -p xxxxxx -g
1024x720 -D -r sound:<span class="kw3">local</span>
```

```
-u 后面的是用户名
-p 后面的是密码
-r sound:local 是启用声音
-D 是消除远程终端窗口的边框
-g 设置远程桌面的分辨率为1024x720
```

KVM 快速键

- Ctrl+Alt：在 KVM 中释放或使用键盘/鼠标。
- Ctrl+Alt+f：切换全屏模式。

KVM 快速克隆镜像

```
<span class="co0">#suspend newvm</span>

<span class="co0">#virt-clone --connect=qemu:///system -o vm1 -n vm2
--file=/home/xxx/vms/vm2.img --force --print-xml -d</span>

<span class="co0">#resume newvm</span>
```

KVM 疑难解答

trying to set up timer as Virtual Wire IRQ

如果启动引导镜像位于FAT或NTFS文件系统，可能也会引发这个问题，最好建议镜像位于linux分区中。

方法一

编辑grub启动配置文件，添加

```
noapic nolapic nomce
```

方法二

开机进入BIOS配置，关闭APIC高级选项。不过，这种做法有可能会影响多系统中的其它系统。

could not initialize alarm timer

去除 -clock rtc 或者

检查运行用户是否为 root

VMware (VMDK)、VirtualBox (VDI) 转换成 Qemu COW2 (qcow2) 格式

```
    <span class="co0">#Converting VMDK to Qemu image</span>
    qemu-img convert -f vmdk myimage.vmdk -O qcow2 myimage.qcow2

    <span class="co0">#Converting VDI to Qemu image</span>
    <span class="kw2">wget</span> http:<span
class="sy0">//</span>dhiru.kholia.googlepages.com<span
class="sy0">/</span>vditool.<span class="kw2">tar</span> <span
class="br0">(</span><span class="kw1">for</span> AMD64 systems<span
class="br0">)</span>
    <span class="kw2">sudo</span> <span class="kw2">tar</span> xf
vditool.<span class="kw2">tar</span> -C <span class="sy0">/</span>usr<span
class="sy0">/</span>lib<span class="sy0">/</span>virtualbox
    <span class="re2">LD_LIBRARY_PATH=</span><span
class="sy0">/</span>usr<span class="sy0">/</span>lib<span
class="sy0">/</span>virtualbox <span class="sy0">/</span>usr<span
class="sy0">/</span>lib<span class="sy0">/</span>virtualbox<span
class="sy0">/</span>vditool COPYDD ~<span
class="sy0">/</span>.VirtualBox<span class="sy0">/</span>VDI<span
class="sy0">/</span>myimage.vdi myimage.raw

    qemu-img convert -f raw myimage.raw -O qcow2 myimage.qcow2
```

挂载 RAW image file

```
    qemu-img convert -f qcow2 myimage.qcow2 -O raw myimage.raw
    <span class="kw2">mount</span> -o loop,<span
class="re2">offset=</span><span class="nu0">32256</span> myimage.raw <span
class="sy0">/</span>mnt
```

由VMWare转KVM格式

- http://www.wanleung.com/blog/archives/900

如何禁用virbr0网卡

virbr0是用来做NAT转换的虚拟网卡，在有些环境下是不需要NAT的。因此，可以关闭virbr0

显示已经激活的网卡

```
<span class="co0"># virsh net-list</span>

Name                 State      Autostart
-----------------------------------------
default              active     <span class="kw2">yes</span>
```

禁用默认的网卡

```
<span class="co0"># virsh net-destroy default</span>
<span class="co0"># virsh net-undefine default</span>
<span class="co0"># service libvirtd restart</span>
<span class="co0"># ifconfig</span>
```

CentOS EL/KVM部署实战

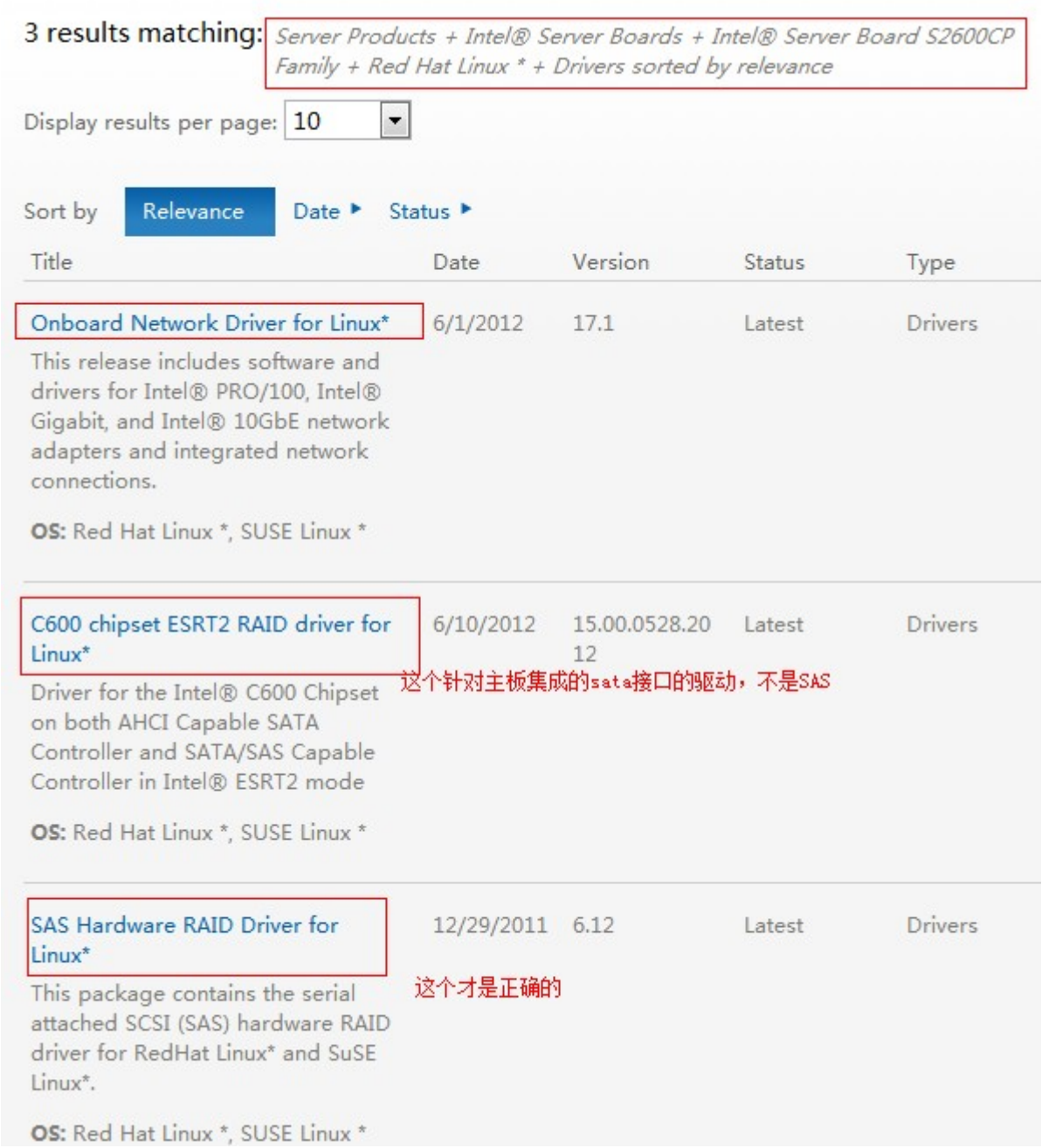
服务器硬件配置

硬件名称	型号	数量	单价
CPU	XEON 2620+1U 散热器	2	2700
硬盘	ST 300G SAS 15K.7	4	1150
内存	创见 8G DDR3 ECC 1333MHZ	8	440
主机板	INTEL S2600CP4/RKSASR5	1	4380

BIOS/RAID配置

```
SAS控制器 -> MegaSAS控制器
MegaSAS配置为 Raid 5
Intel Virtualation Technique(如果不支持升级固件)
```

Intel 主板及raid驱动



搜索关键词"INTEL S2600CP4 raid driver download"

根据INTEL S2600CP4主板型号，先下载相应的raid驱动，通过**安装光盘的dd方式加载**

CentOS EL6安装细节

CentOS EL6分区规划

分区挂载点	文件系统	大小
/boot	ext4	200M
swap	ext4	2000M

/	ext4	5G
/var/lib/libvirt/images	xfs	100G

剩余空间保留，系统安装后再分割裸分区供虚拟机挂载

CentOS 软件包选择

```
Desktop Envirnoments -> Gnome Desktop Envirnoment
Virtualization -> KVM (KVM里面的所有包全选)
```

使用VNC来远程连接安装

```
#vi /etc/libvirt/qemu.conf
#vnc_listen = "0.0.0.0"
#/etc/rc.d/init.d/libvirtd start

virt-install   \
  --connect qemu:///system --virt-type=kvm --accelerate --noapic --noacpi \
  --name ubuntu12 --vcpus=1 --ram 512 --os-type=linux --os-variant=virtio26
\
  --network bridge=br0 --cdrom=/root/ubuntu-12.10-server-amd64.iso --force \
  --graphics type=vnc,listen=192.168.0.123,port=5901  \
  --disk
path=/var/lib/libvirt/images/ubuntu12.img,bus=virtio,size=10,io=native,format
=qcow2,cache=writethrough
```

修改Grub使用KVM

安装之后需要修改/boot/grub/menu.lst文件，因为默认为：

```
Red Hat Enterprise Linux Server (2.6.18-164.el5xen)
```

将开机默认加载改为：

```
Red Hat Enterprise Linux Server-base (2.6.18-164.el5)
```

创建桥接设备

/etc/sysconfig/network-scripts/ifcfg-br0

```
DEVICE=br0
```

```
BOOTPROTO=static
TYPE=Bridge
IPADDR=192.168.1.238
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
ONBOOT=yes
DELAY=0
```

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
BRIDGE=br0
MTU=9000
```

配置后台服务

> - 关闭不必要的服务
> - 关闭防火墙
> - 关闭SElinux
> - 开启 vncserver（因为服务器运行在终端模式下，远程创建虚拟机）

libvirtd 负责KVM进程，必须启动

- **开启 dnsmasq**进程会禁用KVM中的**NAT**模式
- **关闭 messagebus**进程会禁止**haldaemon**启动
- **关闭 haldaemon**进程会禁用KVM中的**Bridge**模式

VIRTIO测试

with virtio_blk (Good!)

```
<span class="br0">[</span>root<span class="sy0">@</span>V1-Geminis-DEV
test1<span class="br0">]</span><span class="co0"># rm -rf 1G;time dd <
/dev/zero > 1G bs=1024 count=1000000</span>
<span class="nu0">1000000</span><span class="nu0">+0</span> records <span
class="kw1">in</span>
<span class="nu0">1000000</span><span class="nu0">+0</span> records out
```

```
real    0m17.366s
user    0m0.154s
sys     0m3.062s
```

without virtio_blk

```
<span class="br0">[</span>root<span class="sy0">@</span>V1-Geminis-DEV <span
class="kw3">test</span><span class="br0">]</span><span class="co0"># rm -rf
1G;time dd < /dev/zero > 1G bs=1024 count=1000000</span>
<span class="nu0">1000000</span><span class="nu0">+0</span> records <span
class="kw1">in</span>
<span class="nu0">1000000</span><span class="nu0">+0</span> records out

real    0m28.559s
user    0m0.194s
sys     0m2.974s
```

with virtio_net (Good!)

client

```
<span class="br0">[</span>root<span class="sy0">@</span>Geminis-DEV ~<span
class="br0">]</span><span class="co0"># iperf -c 192.168.1.244 -p 12345 -w
65535 -t 10</span>
------------------------------------------------------------
Client connecting to <span class="nu0">192.168</span><span
class="nu0">.1</span><span class="nu0">.244</span>, TCP port <span
class="nu0">12345</span>
TCP window <span class="kw2">size</span>:   <span class="nu0">128</span>
KByte <span class="br0">(</span>WARNING: requested <span
class="nu0">64.0</span> KByte<span class="br0">)</span>
------------------------------------------------------------
<span class="br0">[</span>  <span class="nu0">3</span><span
class="br0">]</span> <span class="kw3">local</span> <span
class="nu0">192.168</span><span class="nu0">.1</span><span
class="nu0">.137</span> port <span class="nu0">54216</span> connected with
<span class="nu0">192.168</span><span class="nu0">.1</span><span
class="nu0">.244</span> port <span class="nu0">12345</span>
<span class="br0">[</span> ID<span class="br0">]</span> Interval
Transfer     Bandwidth
<span class="br0">[</span>  <span class="nu0">3</span><span
class="br0">]</span>  <span class="nu0">0.0</span><span
class="nu0">-10.0</span> sec    <span class="nu0">112</span> MBytes  <span
class="nu0">93.6</span> Mbits<span class="sy0">/</span>sec
```

server

```
<span class="br0">[</span>root<span class="sy0">@</span>V1-Geminis-DEV ~<span
class="br0">]</span><span class="co0"># iperf -s -w 65536 -p 12345</span>
------------------------------------------------------------
Server listening on TCP port <span class="nu0">12345</span>
TCP window <span class="kw2">size</span>:   <span class="nu0">128</span>
KByte <span class="br0">(</span>WARNING: requested <span
class="nu0">64.0</span> KByte<span class="br0">)</span>
------------------------------------------------------------
<span class="br0">[</span>  <span class="nu0">4</span><span
class="br0">]</span> <span class="kw3">local</span> <span
class="nu0">192.168</span><span class="nu0">.1</span><span
class="nu0">.244</span> port <span class="nu0">12345</span> connected with
<span class="nu0">192.168</span><span class="nu0">.1</span><span
class="nu0">.137</span> port <span class="nu0">54216</span>
<span class="br0">[</span> ID<span class="br0">]</span> Interval
Transfer     Bandwidth
<span class="br0">[</span>  <span class="nu0">4</span><span
class="br0">]</span>  <span class="nu0">0.0</span><span
class="nu0">-10.0</span> sec    <span class="nu0">112</span> MBytes  <span
class="nu0">93.6</span> Mbits<span class="sy0">/</span>sec
```

without virtio_net

client

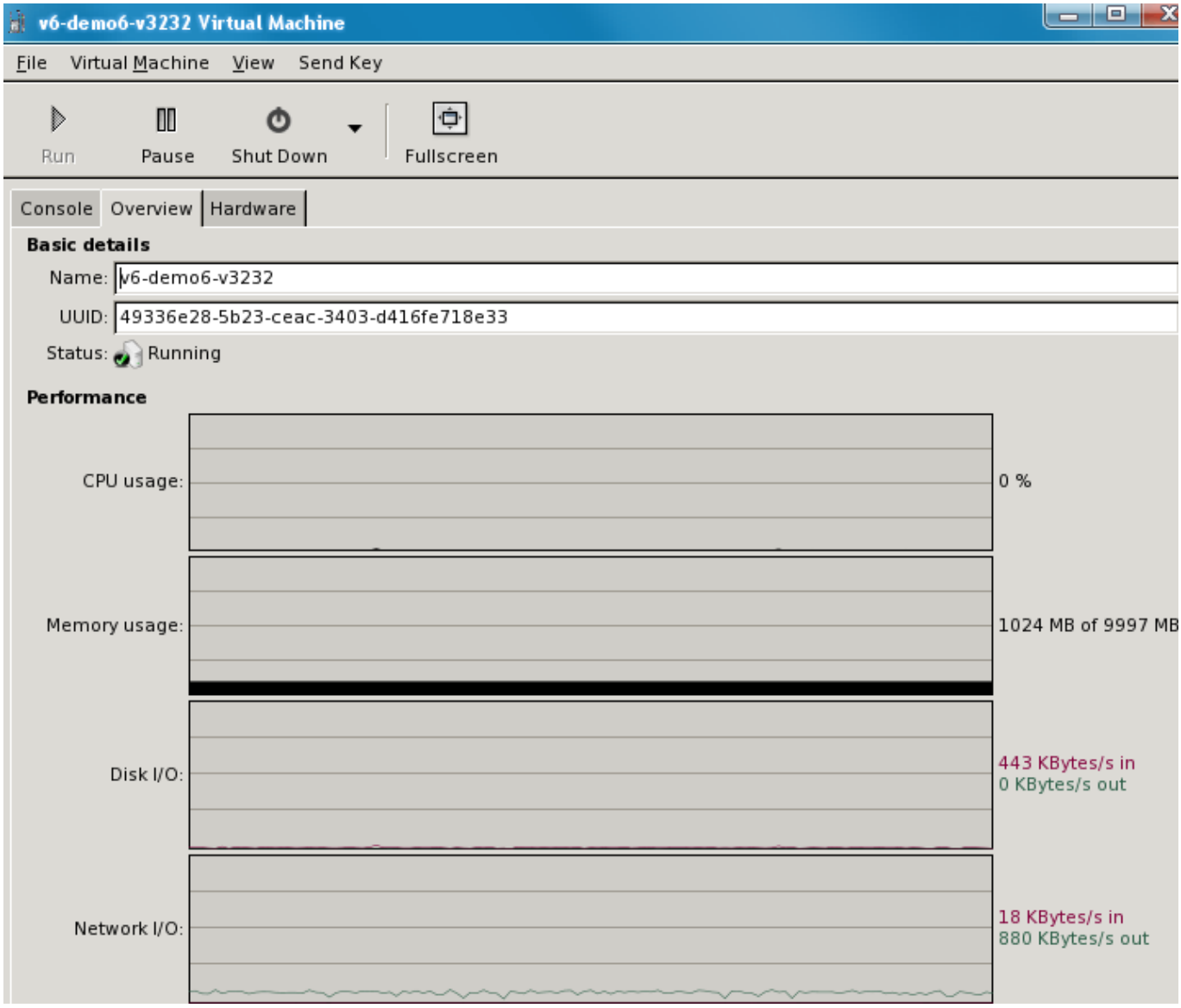
```
-bash<span class="nu0">-3.00</span><span class="co0"># iperf -c
192.168.10.244 -p12345 -w65535 -t 10</span>
------------------------------------------------------------
Client connecting to <span class="nu0">192.168</span><span
class="nu0">.10</span><span class="nu0">.244</span>, TCP port <span
class="nu0">12345</span>
TCP window <span class="kw2">size</span>:   <span class="nu0">128</span>
KByte <span class="br0">(</span>WARNING: requested <span
class="nu0">64.0</span> KByte<span class="br0">)</span>
------------------------------------------------------------
<span class="br0">[</span>  <span class="nu0">3</span><span
class="br0">]</span> <span class="kw3">local</span> <span
class="nu0">192.168</span><span class="nu0">.10</span><span
class="nu0">.1</span> port <span class="nu0">9953</span> connected with <span
class="nu0">192.168</span><span class="nu0">.10</span><span
class="nu0">.244</span> port <span class="nu0">12345</span>
<span class="br0">[</span> ID<span class="br0">]</span> Interval
Transfer     Bandwidth
<span class="br0">[</span>  <span class="nu0">3</span><span
class="br0">]</span>  <span class="nu0">0.0</span><span
```

```
class="nu0">-10.0</span> sec    <span class="nu0">110</span> MBytes  <span
class="nu0">91.9</span> Mbits<span class="sy0">/</span>sec
```

server

```
<span class="br0">[</span>root<span class="sy0">@</span>V1-Geminis-DEV
test1<span class="br0">]</span><span class="co0"># iperf -s -w 65536 -p
12345</span>
------------------------------------------------------------
Server listening on TCP port <span class="nu0">12345</span>
TCP window <span class="kw2">size</span>:   <span class="nu0">128</span>
KByte <span class="br0">(</span>WARNING: requested <span
class="nu0">64.0</span> KByte<span class="br0">)</span>
------------------------------------------------------------
<span class="br0">[</span>  <span class="nu0">4</span><span
class="br0">]</span> <span class="kw3">local</span> <span
class="nu0">192.168</span><span class="nu0">.10</span><span
class="nu0">.244</span> port <span class="nu0">12345</span> connected with
<span class="nu0">192.168</span><span class="nu0">.10</span><span
class="nu0">.1</span> port <span class="nu0">9953</span>
<span class="br0">[</span> ID<span class="br0">]</span> Interval
Transfer     Bandwidth
<span class="br0">[</span>  <span class="nu0">4</span><span
class="br0">]</span>  <span class="nu0">0.0</span><span
class="nu0">-10.0</span> sec    <span class="nu0">110</span> MBytes  <span
class="nu0">91.8</span> Mbits<span class="sy0">/</span>sec
```

网络传输中非常稳定，速率也很理想，有图为证



KVM性能优化

启用KSM/UKSM特性

参考资料

1. UKSM评测
2. How to improve KVM performance by adjusting KSM

Enable KSM - Kernel SamePage Merging

```
echo "1" > /sys/kernel/mm/ksm/run
```

Determining whether to use KSM, or turn it off

If the purpose is to run as many VMs as possible, and performance is not an issue, KSM should be kept running. Which means maximum hardware usage efficiency. But, if a server is running with a relatively small amount of VMs and performance is an issue, KSM should be turned off.

CPU性能调优

> 一个原则，虚拟机CPU默认使用与宿主机相同的CPU类型(qemu -cpu host)，但是你也可以虚拟CPU只支持几种特性，如(qemu -cpu qemu64,+ssse3,+sse4.1,+sse4.2,+x2apic)
>
> cat /proc/cpuinfo | grep flags | uniq

网络性能调优

KVM默认使用网络配置只考虑了兼容性，所以性能绝对不是最佳的。

> 1. 强烈推荐利用宿主机的桥接设备
> 2. 使用KVM virtio网络模型来匹配网络
>
> qemu -net nic,**model=virtio**,mac=... **-net tap**,ifname=...
>
> - Windows XP and 2003 registry settings

调度算法的优化

> 1. 宿主机使用elevator=deadline
> 2. 虚拟机使用elevator=noop
>
> ```
> for f in /sys/block/sd*/queue/scheduler;
> do echo "deadline" > $f;
> done
> ```

磁盘IO调优

1. KVM提供cache=wirtethrough要比cache=writeback(aio=native)安全，但性能稍差
2. 裸设备(RAW)性能要高于qcow2等文件类型
3. 如果是RAW,关闭cache=none(aio=threads),减少数据拷贝和总线流量,)
4. 不要使用默认的IDE模型，建议使用virtio模型

qemu -drive file=/dev/mapper/Guest1,**cache=none,if=virtio**

目前不支持BTRFS，会导致严重的IO堵塞！

Using native AIO with qemu-kvm can cause filesystem corruption with sparse images on EXT4

文件系统的优化

1. EXT4/XFS 性能好于 EXT3
2. 无论宿主机，虚拟机，设置 noatime,nodiratime

要使用 mkfs.xfs 将您的新 XFS 文件系统配置成最佳性能，有两个选项。

第一个这样的选项是 -l size=32m ，它告诉 mkfs.xfs 配置您的文件系统使之拥有一个高达 32 MB 的元数据日志。这通过降低在文件系统处于繁忙使用期间元数据日志将“填满”的可能性而改善了性能。

第二个选项通过告诉 mkfs.xfs 将创建的分配组的数目最小化文件系统的性能。

通常,mkfs.xfs 自动选择分配组的数目，但是，根据我的经验，它通常会选择一个比大多数用于一般用途的 Linux 工作站和服务器过高一点的数目。分配组让 XFS 并行执行多个元数据操作。这为高端服务器带来了便利，但是太多的分配组确实会增加一些开销。因此，不要让 mkfs.xfs 为您的文件系统选择分配组的数目，而是通过使用 -d agcount=x 选项指定一个数目。将 x 设置成一个小数目，如 4、6 或 8。您需要使得您的目标块设备中每 4 GB 容量至少有一个分配组。同时进行这两项调整，使用下面的命令创建“优化的”XFS 文件系统：

```
# mkfs.xfs -d agcount=4 -l size=32m /dev/sdb
```

既然已经创建了文件系统，您就可以挂装它了。挂装时，您将使用一些性能增强 mount 选项来最大程度地发掘出（或发挥出）您的新文件系统的性能。

```
# mount /dev/sdb /mnt -o noatime,nodiratime,osyncisdsync
```

前面的两个 mount 选项关闭 atime 更新，几乎不需要 atime 更新，并且它除了降低文件系统性能之外几乎不起任何作用。 osyncisdsync 选项调整 XFS 的同步／异步行为，以便它同 ext3 更一致。多亏了我们的 mkfs.xfs 和 mount 调整，您的新 XFS 文件系统比没这么调整时的性能要好得多。

Best practice: Optimize block I/O performance by using the Deadline I/O scheduler

参考资料

- kvm网络桥接方案
- Kernel-based Virtual Machine (KVM) 基本操作備忘
- CentOS下KVM虚拟机文字模式安装指令virt-install
- RedHat KVM Installation
- Get a performance boost by backing your KVM guest with hugetlbfs

```
<span class="kw2">grep</span> Huge <span class="sy0">/</span>proc<span
class="sy0">/</span>meminfo
HugePages_Total:       <span class="nu0">0</span>
HugePages_Free:        <span class="nu0">0</span>
HugePages_Rsvd:        <span class="nu0">0</span>
HugePages_Surp:        <span class="nu0">0</span>
Hugepagesize:       <span class="nu0">2048</span> kB
```

Intel VT和 AMD-V 是一套与支持该技术的虚拟机监视器相结合的硬件增强特性(指令集扩展)。

kvm可在原始硬件速度下通过运行完全隔离的虚拟机来执行任务。

- KVM在Ubuntu下的安装
- 虚拟机管理器

APIC: Advanced Programmable Interrupt Controller高级程序中断控制器.

> APIC 是装置的扩充组合用来驱动 Interrupt 控制器。在目前的建置中，系统的每一个部份都是经由 APIC Bus 连接的。
>
> “本机 APIC” 为系统的一部份，负责传递 Interrupt
> 至指定的处理器；举例来说，当一台机器上有三个处理器则它必须相对的要有三个本机 APIC
> 。自 1994 年的 Pentium P54c 开始Intel 已经将本机 APIC 建置在它们的处理器中。实际建置了 Intel 处理器的电脑就已经包含了 APIC 系统的部份。
>
> 系统中另一个重要的部份为 I/O APIC。系统中最多可拥有 8 个 I/O APIC。它们会收集来自 I/O 装置的 Interrupt 讯号且在当那些装置需要 interrupt 时传送讯息至本机 APIC。每个 I/O APIC 有一个专有的 interrupt 输入 (或 IRQ) 号码。Intel 过去与目前的 I/O APIC 通常有 24 个输入 –
> 其它的可能有多達 64 个。而且有些机器拥有数个 I/O APIC
> ，每一个分别有自己的输入号码，加起来一台机器上会有上百个 IRQ 可供装置 Interrupt 使用。

问题跟踪

Cannot find display

```
[root@SA-KVM ~]#
login as: root
root@192.168.0.253's password:
Last login: Wed May 11 20:12:48 2011
/usr/bin/xauth:  creating new authority file /root/.Xauthority
```

yum groupinstall "X Window System"
export DISPLAY=0.0

如果管理界面出现了方块码

```
yum <span class="kw2">install</span> -y libfonts.x86_64
xorg-x11-fonts-Type1.noarch
```

sudo virt-manager 报错的快速解决方法

virt-manager 报以下错误

```
X11 connection rejected because of wrong authentication.
Traceback (most recent call last):
File "/usr/share/virt-manager/virt-manager.py", line 383, in <module>
main()
File "/usr/share/virt-manager/virt-manager.py", line 286, in main
raise gtk_error
RuntimeError: could not open display
```

使用 mkxauth 命令将ssh登陆的用户的 Xauthority 信息加入到 root 用户的/root/.Xauthority 文件中

```
$ssh -X bitbi@mybitbihost
$sudo mkxauth -m bitbi
merging /home/bitbi/.Xauthority into /root/.Xauthority ...
done
$sudo virt-manager
```

下面运行 sudo virt-manager 命令就正常了。

> 或者更换display端口, /etc/profile
>
> ```
> export DISPLAY=localhost:10.0
> ```

HARDWARE ERROR! Processor Context Corrupt

> 1. Hardware Error! - on Linux Fedora 12 install
> 2. Hardware Error! - on Fedora 12 install