本文大纲

一、简介

二、kvm虚拟机安装

三、配置KVM虚拟机运行的网络环境

四、配置kvm运行环境

五、创建guest虚拟机实例

六、使用vnc连接子虚拟机安装系统

七、kvm虚拟机常用命令、快照、克隆迁移、磁盘管理

八、kvm虚拟机优化与配置文件详解

九、Kvm中常见问题与解决

###############

**一、KVM虚拟机简介**

KVM是kernel-based Virtual Machine的简称，是一个内核模块，使用它需要CPU支持虚拟化。加载KVM模块后，系统中会有一个/dev/kvm 设备，这个设备提供 ioctl 和 mmap操作。目前已成为学术界的主流VMM之一。KVM的虚拟化需要硬件支持（如Intel VT技术或者AMD V技术)，是基于硬件的完全虚拟化。KVM的安装和使用相对于XEN来说十分的简单和方便，并且功能强大，比较适用于高性能计算服务的云计算环境中作为IAAS层的虚拟化部署工作。不得不提的是甲骨文公司的VirtualBox也很强大，各方面与KVM可以相媲美，但是现在不开源比较昂贵一般的云计算平台很少使用。至于完全虚拟化和半虚拟化我觉得这是一个很模糊的概念。

KVM提供了图像界面的管理接口(Virtual Machine Manager)和命令行式的管理接口（virsh）。可以根据使用的场景采用不同的方式，当然也可以使用Libvirt库进行管理虚拟机，并且使用libvirt库进行虚拟机的管理是业界很推崇的做法，这都源于libvirt库良好的移植性和强大的API，并且提供了多种语言接口（如C语言，python语言，JAVA语言，C#语言和PHP语言）能对Xen，KVM以及QEMU等多类虚拟机进行管理管理。详情请参考其官网：<http://libvirt.org/。>

一个典型的 KVM 安装包括以下部件

一个管理虚拟硬件的设备驱动，这个驱动通过一个字符设备 /dev/kvm 导出它的功能。通过 /dev/kvm每一个客户机拥有其自身的地址空间，这个地址空间与内核的地址空间相分离或与任何一个正运行着的客户机相分离。

一个模拟硬件的用户空间部件，它是一个稍微改动过的 QEMU 进程。从客户机操作系统执行 I/O 会拥有QEMU 。 QEMU 是一个平台虚拟化方案，它允许整个 PC 环境（包括磁盘、显示卡（图形卡）、网络设备）的虚拟化。任何客户机操作系统所发出的 I/O 请求都被拦截，并被路由到用户模式用以被 QEMU 过程模拟仿真。

**二、kvm虚拟机安装**

**前期的准备**

**2.1kvm的诞生就需要硬件虚拟化扩展，物理机必须支持虚拟化，确保BIOS里面开启了vt，KVM 需要有 CPU 的支持（Intel VT 或 AMD SVM），在安装 KVM 之前检查一下 CPU 是否提供了虚拟技术的支持**

cat /proc/cpuinfo查看cpu是否支持虚拟化如果看到输出中有pae，那么cpu 就支持半虚拟化，如果输出中同时有pae和vmx/svm(注:vmx为intel,svm为amd)，那么cpu支持全虚拟

Uname -r 查看linux内核版本如果低于2.6.20以下的linux版本需要升级内核

**2.2使用yum安装KVM虚拟机**

2.2.1使用yum在线安装KVM的主要组件及工具：

第一种方式安装所有相关组件：

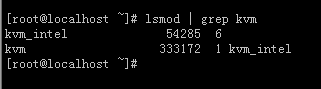
yum install -y kvm kmod-kvm kvm-qemu-img libvirt Python-virtinst virt-manager virt-viewer bridge-utils

第二种方式安装virtualization工具包：

yum groupinstall "Virtualization" "Virtualization Client" "Virtualization Tools" "Virtualization Platform"

不管哪种方式安装完成以后reboot在查看linux是否已经加载kvm模块

lsmod | grep kvm 显示结果如下表示kvm模块已经加载，如果没有输出信息，则表示没有加载成功



2.2.2libvirt简介

Libvirt是目前使用最广泛的虚拟机管理工具和应用程序接口（API），常用的虚拟机管理工具（virsh、virt-install、virt-manager）和云计算架构平台（如：OpenStack、OpenNubula）等在底层都使用libvirt的应用程序接口。

Libvirt是一套开源的支持linux下虚拟化工具的函数库，为安全有效的管理各域，libvirt提供的一个公共的稳定软件层。

Libvirt由三个部分组成：应用程序接口（API）、一个守护进程libvirtd、一个默认的命令行管理工具virsh。

作用分别如下：

API为其他虚拟机管理工具（如virsh virt-manager）提供程序库支持；

Libvirtd守护进程管理所有域，在使用各工具对虚拟机进行管理时，该守护进程需要打开；

Virsh是libvirt项目中默认的命令行管理工具，具有定义、启动、停止等管理功能。

管理工具virsh、virt-manager、virt-install等，通过使用libvirt提供的API的，对虚拟化程序（Hypervisor）在各物理节点（node）上虚拟化出的多个域（domain，机客户操作系统 Guest OS）进行操作管理。

**三、配置KVM虚拟机运行的网络环境**

**安装网络桥接工具yum install bridge-utils tunctl -y**

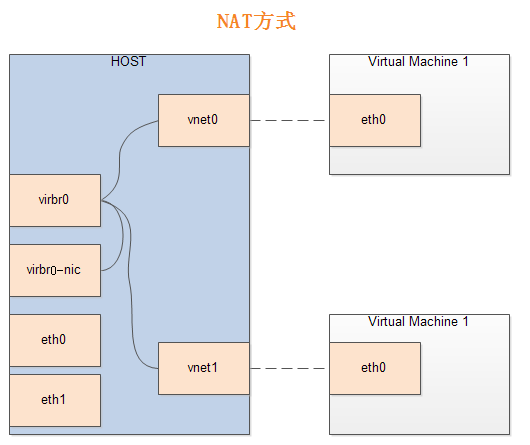
**执行brctl show命令查看host桥接状态**

1说明：KVM默认是使用NAT模式为虚拟机（客户操作系统）提供网络的，还有一种网络模式叫做桥接模式，建议使用桥接模式

1.1nat模式配置

KVM默认采用nat模式，用户网络（User Networking）：让虚拟机访问主机、互联网或本地网络上的资源的简单方法，但是不能从网络或其他的客户机访问客户机。在公网IP不够使用KVM还需要上网的时候可以使用，大大节省了公网IP！同时这种模式也使得KVM不用暴露在公网之中，也增加了安全性

下图是虚拟机管理模块产生的接口关系：



其中virbr0是由宿主机虚拟机支持模块安装时产生的虚拟网络接口，也是一个switch和bridge，负责把内容分发到各虚拟机。从图上可以看出，虚拟接口和物理接口之间没有连接关系，所以虚拟机只能在通过虚拟的网络访问外部世界，无法从网络上定位和访问虚拟主机。

宿主机配置

cp /etc/libvirt/qemu/networks/default.xml /etc/libvirt/qemu/networks/default.xml.bk

cat /etc/libvirt/qemu/networks/default.xml

<network>

<name>default</name> //网络名称

<uuid>ea07af3d-fd95-444d-a1d2-a0ae5fce43de</uuid> ///修改唯一标识符

<forward mode='nat'/> //此处增加mode参数为nat

<bridge name='virbr0' stp='on' delay='0' />

<mac address='52:54:00:89:DD:12'/>

<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>

内部网络的管理——定义、启动、关闭等

virsh net-define /etc/libvirt/qemu/networks/default.xml

virsh net-destroy default

[virsh net-start default

virsh net-autostart natwork //设置指定内部网络为开启启动

ls /etc/libvirt/qemu/networks/autostart/

default.xml natwork.xml

//设置开机启动的内部网络，会自动在autostart下生成配置链接

cat vm.xml

······

<interface type='network'>

<mac address='52:54:00:66:6e:49'/>

<source network='default'/>

<model type='virtio'/> //此处为新增参数mode

<address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>

</interface>

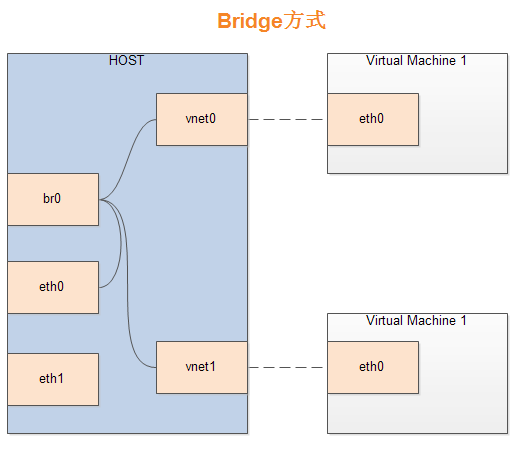
······

1.2Bridge方式

虚拟网桥（Virtual Bridge）：设置好后客户机与互联网，客户机与主机之间都可以通信，试用于需要多个公网IP的环境。

Bridge方式即虚拟网桥的网络连接方式，是客户机和子网里面的机器能够互相通信。可以使虚拟机成为网络中具有独立IP的主机。

桥接网络（也叫物理设备共享）被用作把一个物理设备复制到一台虚拟机。网桥多用作高级设置，特别是主机多个网络接口的情况



如上图，网桥的基本原理就是创建一个桥接接口br0，在物理网卡和虚拟网络接口之间传递数据。

配置kvm虚拟机桥接网卡

cp ifcfg-eth0 ifcfg-br0

将ifcfg-br0修改为如下，根据不同的机器与ip需要修改不同的地址

DEVICE=br0

TYPE=Bridge

ONBOOT=yes

BOOTPROTO=static

IPADDR=192.168.24.230

NETMASK=255.255.255.0

GATEWAY=192.168.24.254

DNS1=114.114.114.114

DELAY=0

############################

修改ifcfg-eth0为

DEVICE=eth0

HWADDR=94:DE:80:5B:AF:3A

TYPE=Ethernet

UUID=6346d8b1-f6e5-4913-a811-ed85134f9cda

ONBOOT=yes

BRIDGE=br0

#########################

修改以后重启网卡

server network restart

Route 一下查看网关路由信息等

Ping 一下外网如果能通表示桥接配置成功

**四、配置kvm运行环境**

关闭selinux虚拟机kvm不支持开启selinux

vi /etc/sysconfig/selinux --------SELINUX=disabled

vi /etc/libvirt/qemu.conf -------vnc\_listen = "0.0.0.0"修改vnc监听端口，

如果使用vnc连接虚拟机必须改了0.0.0.0

关闭防火墙或者开放访问需要的端口

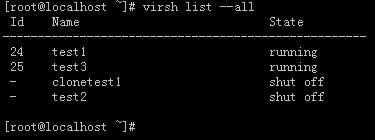
Service iptables stop

重启一下系统让其生效reboot

开启libvirtd用来管理kvm虚拟机

/etc/init.d/libvirtd restart

virsh list --all 查看以下虚拟机 会显示Id,名称,状态 ，如下如果没有虚拟机则为空



Kvm虚拟机搭建以后，使用libvirtd管理，开启程序以后首先需要创建子虚拟机，不然virsh list查看的虚拟机为空，子虚拟机创建以后使用vnc连接，然后按照系统配置环境。

**五、创建guest虚拟机实例：**

5.1创建一块磁盘

qemu-img create -f qcow2 -o size=20G /var/xen\_disk/test1.qcow2

创建成功以后可以使用qemu-img info查看磁盘信息，kvm磁盘格式中只有qcow2格式支持创建虚拟机快照所以建议使用qcow2格式。

qemu-img info /var/xen\_disk/test1.qcow2

5.2使用virt-install创建子虚拟机下面是一个实例

通过本地光盘安装——网络为桥接模式

virt-install \

--virt-type kvm \

--name test1 \

--ram 1024 \

-vcpus=1 \

--network bridge=br0 \

--accelerate \

--vnc --vncport=5903 --vnclisten=0.0.0.0 \

--hvm --os-variant=rhel6 \

--disk path=/var/xen\_disk/test3.qcow2,size=20,format=qcow2 \

--cdrom=/var/system\_iso/CentOS-6.5-x86\_64-bin-DVD1.iso \

--os-type=linux \

--noautoconsole

通过网络安装——网络为桥接模式

virt-install \

--virt-type kvm \

--name test3 \

--ram 1024 \

-vcpus=1 \

--network bridge=br0 \

--accelerate \

--vnc --vncport=5903 --vnclisten=0.0.0.0 --passwd=123456 \

--hvm --os-variant=rhel6 \

--disk path=/var/xen\_disk/test3.qcow2,size=20,format=qcow2 \

--location=ftp://192.168.24.234/centos6.5 \

--os-type=linux \

--noautoconsole

--name 指定虚拟机名称，virsh操作指定虚拟机时所需要的参数，不可以重复。

--ram 分配内存大小，安装完成后可以用 virsh 调整。

--vcpus 分配CPU核心数，最大与实体机CPU核心数相同，安装完成后也可以用 virsh 调整。

<vcpu placement='static' cpuset='2-3'>2</vcpu>#cpuset可以

--disk 指定虚拟机镜像， size 指定分配大小单位为G。

--network 网络类型，此处用的是默认，一般用的应该是 bridge 桥接。

--os-variant 指定操作系统类型，此处使用的是标准Linux 2.6，其他的可以通过 man virt-install 详细查看。

--accelerate 加速，具体什么原理还不太清楚。

--cdrom 指定安装镜像所在。

--vnc 启用VNC远程管理，一般安装系统都要启用。

--vncport 指定 VNC 监控端口，默认端口为 5900，端口不能重复。

--vnclisten 指定 VNC 绑定IP，默认绑定127.0.0.1，这里将其改为 0.0.0.0 以便可以通过外部连接

--location 使用nfs:host/path、ftp://host/path、http://host/path的方式指定安装源

--noautoconsole #不自动开启控制台

创建以后虚拟机是未被开启的需要执行

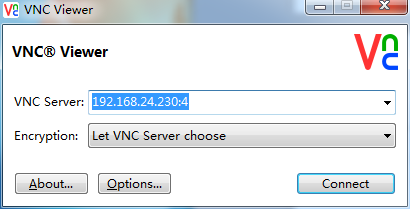
**Virsh start kvm #启动虚拟机**

虚拟机镜像的默认位置在/var/lib/libvirt/images/文件夹下，如果安装时候指定了就在指定目录下面。

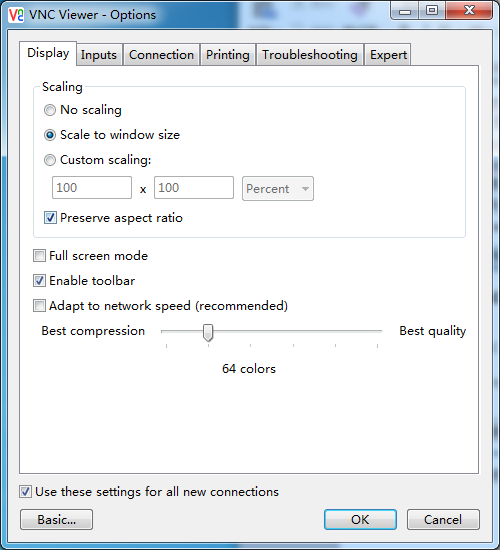
虚拟机的xml配置文件默认在/etc/libvirt/qemu/目录下。

**六、使用vnc连接子虚拟机安装系统**

6.1**在windows下面下载一个RealVNC.xp510.com安装并打开**



在选项里面将best compression减低一些否则会直接闪退



回到首页在vnc server里面填写母机的ip地址+子机vnc配置的端口点击连接

连接成功就会出现虚拟机安装界面

**6.2远程管理虚拟机（qemu+ssh连接）**

export DISPLAY=192.168.40.18:0.0 #修改一下环境变量里面的display不然无法显示图形界面

virt-viewer -c qemu:///system node4   #本地管理虚拟机，system：获取system权限，注意qemu后是三个/

virt-viewer -c qemu+ssh://root@192.168.32.40/system node4  #远程linux通过virt-viewer+ssh管理虚拟机

Xlib:  extension "RANDR" missing on display "192.168.40.18:0.0".

root@192.168.32.40's password:

root@192.168.32.40's password:

这个密码就是当初设置vnc访问密码

#会弹出virt-viwer的gtk管理界面

第一次连接guest以后会出现linux系统安装界面，安装步骤不做详解了，最后系统选择mini版再加上Development tools工具包，安装系统要求尽量简洁。

**七、kvm虚拟机常用命令、快照、克隆迁移、磁盘管理**

**7.1Kvm虚拟机常用命令**

virsh list 列出当前虚拟机列表，不包括未启动的

virsh list --all 列出所有虚拟机，包括所有已经定义的虚拟机

virsh start domain-name 启动指定虚拟机

virsh shutdown domain-name 停止指定虚拟机

virsh destroy domain-name 强制停止指定虚拟机

virsh reboot domain-name 重新启动指定虚拟机

virsh autostart domain-name 指定虚拟机开机自动启动

virsh autostart --disable domain-name 关闭指定虚拟机开机自动启动

virsh vncdisplay domain-name 查看该虚拟机vnc开放的端口,以便使用vnc连接该虚拟机

virsh suspend oeltest01 挂起服务器

virsh resume oeltest01 恢复服务器

virsh undefine centos08 --managed-save 删除一个虚拟机子机(请谨慎操作)

删除一个虚拟机会删除/etc/libvirt/qemu/这个目录的配置文件，虚拟机磁盘需要手动删除

**7.2Kvm虚拟机克隆**

为了更快速度的搭建一个相同环境的kvm子虚拟机可以使用kvm虚拟机克隆技术。

克隆一个虚拟机命令virt-clone

下面的实例是讲kvm1克隆为kvm2很简单，只要制定新的虚拟机名字和磁盘地址就可以了，命令执行以后会自动生成配置文件，并且定义好新虚拟机。

virt-clone -o kvm1 -n kvm2 -f /var/xen\_disk/kvm2.qcow2

配置文件默认在/etc/libvirt/qemu/下面。

克隆成功以后会生成磁盘文件,与虚拟机配置文件.因为这个两个文件都和克隆主机一样所以需要修改一些参数不然无法同时开机。

执行virsh edit newkvm

1如果正常情况一般uuid或mac地址会在克隆以后生成新的，如果没有被修改，那么需要修改配置文件中的mac和uuid

<uuid>5a448cb5-99d9-c842-8542-c5269870e512</uuid>

<mac address='52:54:00:a5:48:db'/>

2vnc端口（如果想将vnc密码改成不一样也可以修改）

<graphics type='vnc' port='5903' autoport='no' listen='0.0.0.0' passwd=’123456’ >

3开启克隆后的虚拟机配置新的网络ip地址

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

Vi /etc/sysconf/network 修改主机名

#修改dns配置

cat >/etc/resolv.conf <<EOF

search localdomain

nameserver 202.96.128.86

nameserver 202.96.128.166

nameserver 114.114.114.114

nameserver 8.8.8.8

EOF

4重启新虚拟机reboot

**7.3kvm虚拟机创建快照**

**注意:要使用 KVM 的快照功能，虚拟机的硬盘一定要是 qcow2 格式，否则无法使用快照功能**

virsh snapshot-create-as kvm1 kvm1xxxx //创建一个虚拟机快照

virsh　snapshot-list　kvm1 //查看虚拟机快照

virsh snapshot-revert kvm1 kvm1xxxx //恢复kvm1虚拟机的状态到kvm1xxxx快照

virsh snapshot-delete kvm1 kvm1xxxx //删除一个虚拟机快照

**7.4KVM虚拟机磁盘镜像文件管理命令qemu-img介绍及简单使用**

qemu-img是QEMU的磁盘管理工具，在qemu-kvm源码编译后就会默认编译好qemu-img这个二进制文件。qemu-img也是QEMU/KVM使用过程中一个比较重要的工具

1 check [-f fmt] filename

对磁盘镜像文件进行一致性检查，查找镜像文件中的错误，目前仅支持对“qcow2”、“qed”、“vdi”格式文件的检查。

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

还可以添加一个或多个选项（options）来附加对该文件的各种功能设置，可以使用“-o ?”来查询某种格式文件支持那些选项，在“-o”选项中各个选项用逗号来分隔。

3 info [-f fmt] filename

展示filename镜像文件的信息。如果文件是使用稀疏文件的存储方式，也会显示出它的本来分配的大小以及实际已占用的磁盘空间大小。如果文件中存放有客户机快照，快照的信息也会被显示出来。

4 convert [-c] [-p] [-f fmt] [-t cache] [-O output\_fmt] [-o options] [-s snapshot\_name] [-S sparse\_size] filename [filename2 [...]] output\_filename

将fmt格式的filename镜像文件根据options选项转换为格式为output\_fmt的名为output\_filename的镜像文件。它支持不同格式的镜像文件之间的转换，比如可以用VMware用的vmdk格式文件转换为qcow2文件，这对从其他虚拟化方案转移到KVM上的用户非常有用。一般来说，输入文件格式fmt由qemu-img工具自动检测到，而输出文件格式output\_fmt根据自己需要来指定，默认会被转换为与raw文件格式（且默认使用稀疏文件的方式存储以节省存储空间）

5 snapshot [-l | -a snapshot | -c snapshot | -d snapshot] filename

“-l” 选项是查询并列出镜像文件中的所有快照，“-a snapshot”是让镜像文件使用某个快照，“-c snapshot”是创建一个快照，“-d”是删除一个快照。

6 rebase [-f fmt] [-t cache] [-p] [-u] -b backing\_file [-F backing\_fmt] filename

改变镜像文件的后端镜像文件，只有qcow2和qed格式支持rebase命令。使用“-b backing\_file”中指定的文件作为后端镜像，后端镜像也被转化为“-F backing\_fmt”中指定的后端镜像格式。

它可以工作于两种模式之下，一种是安全模式（Safe Mode）也是默认的模式，qemu-img会去比较原来的后端镜像与现在的后端镜像的不同进行合理的处理；另一种是非安全模式（Unsafe Mode），是通过“-u”参数来指定的，这种模式主要用于将后端镜像进行了重命名或者移动了位置之后对前端镜像文件的修复处理，由用户去保证后端镜像的一致性。

7 resize filename [+ | -]size

改变镜像文件的大小，使其不同于创建之时的大小。“+”和“-”分别表示增加和减少镜像文件的大小，而size也是支持K、M、G、T等单位的使用。缩小镜像的大小之前，需要在客户机中保证里面的文件系统有空余空间，否则会数据丢失，另外，qcow2格式文件不支持缩小镜像的操作。在增加了镜像文件大小后，也需启动客户机到里面去应用“fdisk”、“parted”等分区工具进行相应的操作才能真正让客户机使用到增加后的镜像空间。**不过使用resize命令时需要小心（最好做好备份），如果失败的话，可能会导致镜像文件无法正常使用而造成数据丢失（增加以后需要关闭虚拟机，再开启虚拟机，重启不生效）**

**qemu-img的使用**

1创建磁盘

qemu-img create -f qcow2 -o size=20G /var/xen\_disk/test1.qcow2

2查看虚拟机镜像文件

qemu-img info /var/xen\_disk/test1.qcow2

查看磁盘信息是发现这个磁盘实际大小是0因为他是稀疏文件，只有被真正使用以后才会增加大小。

3转换磁盘格式将qemu-img支持的格式如下所示：

vvfat vpc vmdk vdi sheepdog rbd raw host\_cdrom host\_floppy host\_device file qed qcow2 qcow parallels nbd dmg tftp ftps ftp https http cow cloop bochs blkverify blkdebug

以下命令是把文件从qcow2转成raw格式，然后再查看新镜像文件。

qemu-img convert -f qcow2 test\_lvm.qcow2 -O raw test\_lvm.raw

4改变镜像大小，**这个命令执行前需要备份好磁盘数据，以免失败导致数据丢失**

qemu-img resize test.raw +1GB

给次磁盘增加1G的大小，**注意raw才支持resize**

5磁盘镜像快照，建议使用virsh snapshot-create-as创建，这里跳过

快照这个功能只支持qcow2格式，raw不支持。其中-c是创建快照，-l是查看快照，-a是恢复快照，-d是删除快照。注意，创建磁盘快照时虚拟机需要处理关闭的状态。

**八、kvm虚拟机优化与配置文件详解**

**8.1导出kvm虚拟机配置文件**

#virsh dumpxml wintest01 > /etc/libvirt/qemu/wintest02.xml

**8.2Kvm虚拟机配置文件**

每一个kvm虚拟机安装以后至少有一个配置文件和磁盘文件，kvm虚拟机移植起来也非常的方便只要将这两个文件移到其他主机就可以重新定义启动。

配置文件默认存放在/etc/libvirt/qemu/目录下面，配置文件记录了该虚拟机的所有配置信息，虚拟机每次启动都会读取配置文件里面的信息。

配置文件可以使用virsh edit kvm.xml修改或者使用vi修改，vi修改以后需要重启libvirtd才能生效，kvm虚拟机配置文件修改不建议使用vi直接编辑xml配置文件，推荐使用virsh edit kvm.xml编辑配置文件，编辑完以后重启虚拟机生效。

**下面是kvm虚拟机配置文件详解：**

<domain type='kvm'>

domain 是一个所有虚拟机都需要的根元素，它有两个属性，type定义使用哪个虚拟机管理程序，值可以是：xen、kvm、qemu、lxc、kqemu，第二个参数是id，它唯一的标示一个运行的虚拟机，不活跃的客户端没有id

<name>kvm\_test3</name>

name参数为虚拟机定义了一个简短的名字，必须唯一

<uuid>f7333079-650e-8bea-4c36-184480afa0ba</uuid>

uuid为虚拟机定义了一个全球唯一的标示符，uuid的格式必须遵循RFC 4122指定

的格式，当创建虚拟机没有指定uuid时会随机的生成一个uuid

<boot dev='hd'/>

dev属性的值可以是：fd、hd、cdrom、network，它经常被用来指定下一次启动。boot的元素可以被设置多个用来建立一个启动优先规则

CPU分配：

<vcpu placement='static' cpuset="1-4,^3,6" current="1">2</vcpu>

vcpu的内容是为虚拟机最多分配几个cpu，值处于1~maxcpu之间，可选参数：cpuset参数指定虚拟cpu可以映射到那些物理cpu上，物理cpu用逗号分开，单个数字的标示单个cpu，也可以用range符号标示多个cpu，数字前面的脱字符标示排除这个cpu，current参数指定虚拟机最少，placement参数指定一个domain的cpu的分配模式，值可以是static、auto。

内存分配：

内存大小为1024\*1024\*n，n是以G为单位

<memory unit='KiB'>524288</memory>

memory 定义客户端启动时可以分配到的最大内存，内存单位由unit定义，单位可以是：K、KiB、M、MiB、G、GiB、T、TiB。默认是KiB。

<currentMemory>1024000</currentMemory>

currentMemory 定义实际分给给客户端的内存她小于memory的定义，如果没有定义，值和memory一致。

控制周期：

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<on\_crash>restart</on\_crash>

当一个客户端的OS触发lifecycle时，它将采取新动作覆盖默认操作，具体状态参

数如下：

on\_poweroff：当客户端请求poweroff时执行特定的动作

on\_reboot：当客户端请求reboot时执行特定的动作

on\_crash：当客户端崩溃时执行的动作

每种状态下可以允许指定如下四种行为：

destory：domain将会被完全终止，domain的所有资源会被释放

restart：domain会被终止，然后以相同的配置重新启动

preserver：domain会被终止，它的资源会被保留用来分析

rename-restart：domain会被终止，然后以一个新名字被重新启动

当一个客户端的OS触发lifecycle时，它将采取新动作覆盖默认操作，具体状态参数如下：

on\_poweroff：当客户端请求poweroff时执行特定的动作

on\_reboot：当客户端请求reboot时执行特定的动作

on\_crash：当客户端崩溃时执行的动作

每种状态下可以允许指定如下四种行为：

destory：domain将会被完全终止，domain的所有资源会被释放

restart：domain会被终止，然后以相同的配置重新启动

preserver：domain会被终止，它的资源会被保留用来分析

rename-restart：domain会被终止，然后以一个新名字被重新启动

Hypervisor的特性：

<features>

<pae/>

<acpi/>

<apic/>

<hap/>

<privnet/>

</features>

Hypervisors允许特定的CPU/机器特性打开或关闭，所有的特性都在fearures元素中，以下介绍一些在全虚拟化中常用的标记：

pae：扩展物理地址模式，使32位的客户端支持大于4GB的内存

acpi：用于电源管理

hap：Enable use of Hardware Assisted Paging if available in the hardware.

时间设置：

<clock offset="localtime" />

客户端的时间初始化来自宿主机的时间，大多数操作系统期望硬件时钟保持UTC格式，UTC也是默认格式，然而Windows机器却期望它是'localtime'

clock的offset属性支持四种格式的时间：UTC localtime timezone variable

UTC：当引导时客户端时钟同步到UTC时钟

localtime：当引导时客户端时钟同步到主机时钟所在的时区

timezone：The guest clock will be synchronized to the requested timezone using the timezone attribute.

设备设置:

<devices>

所有的设备都是一个名为devices元素的子设备(All devices occur as children of the main devices element.)，以下是一个简单的配置：

<emulator>/usr/bin/kvm</emulator>

emulator元素指定模拟设备二进制文件的全路径

<disk type='block' device='disk'>

<driver name='qemu' cache='none'/>

<source dev='/dev/cciss/c0d0p6'/>

<target dev='vda' bus='virtio'/>

</disk>

<disk type='block' device='cdrom'>

<target dev='hdc' bus='ide'/>

<readonly/>

</disk>

所有的设备看起来就像一个disk、floppy、cdrom或者一个 paravirtualized driver，他们通过一个disk元素指定。

disk是一个描述disks的主要容器，type特性包括:file,block,dir,network。device描述disk如何受到客户端OS的，特性包括：floppy、disk、cdrom、lun，默认是disk。snapshot属性表明默认行为在磁盘做snapshot的时候，snapshot的参数有：internal ，在snapshot的时候可以存储改变的数据。external，在snapshot时分开活动的数据。no，disk不参加snapshot，只读磁盘默认是no。

source元素：在disk的type是file时，file属性指定一个合格的全路径文件映像作为客户端的磁盘，在disk的type是block时，dev属性指定一个主机设备的路径作为disk，在disk的type是dir时，dir属性指定一个全路径的目录作为disk，在disk的type是network时，protocol属性指定协议用来访问镜像，镜像的值可以是：nbd，rbd，sheepdog。当protocal的属性值是rbd或者sheepdog时，必须用一个额外的name属性指定使用那个镜像，当type的值是network时，source可以有0个或者多个host字属性指定连接哪些主机。

target元素：控制总线设备在某个磁盘被选为客户端的OS时，dev属性表明本地磁盘在客户端上的实际名称，因为实际设备的名称指定并不能保证映射到客户端OS上的设备。bus属性指定了哪种类型的磁盘被模拟，值主要有：ide、scsi、virtio、xen、usb、sata。如果省略，总线类型从设备名来推断，例如设备名是sda，则使用scsi类型的总线。tray属性指定可移动磁盘的状态，例如cdrom或者floppy，它的值是open或closed，默认是closed。

driver允许更进一步的指定hypervisor driver的相关细节。如果hypervisor支持多个后端驱动程序，name属性选择一个主要的后端驱动的名称，可选type参数可以指定一个子类型，例如：xen支持的名称包括tap、tap2、phy、file，qemu只支持qemu名称，但是多类型的包括raw、bochs、qcow2、qed等。cache属性控制cache机制，值可以是：default、none、writethtough、writeback、directsync、unsafe。error\_policy属性指定当hypervisor在读写磁盘出现错误时的行为，值可以是：stop、report、ignored、enospace，默认值是report。io属性控制IO策略，qemu客户端支持threads、native。

readonly元素：指定客户端不能修改设备。当一个disk含有type=cdrom，readonly则是默认值。

host元素：有两个属性name和port，分别指定了hostname和port。

网络接口：

有好几种网络接口访问客户端:Virtual network、Bridge to LAN、Userspace SLIRP stack、Generic ethernet connection、Direct attachment to physical interface。

Virtual network：这种推荐配置一般是对使用动态/无线网络环境访问客户端的情况。

Bridge to LAN：这种推荐配置一般是使用静态有限网络连接客户端的情况。

<interface type='bridge'>

<source bridge='br0'/>

<mac address='52:54:00:ad:82:97'/>

<model type='virtio'/>

</interface>

输入设备：

输入设备允许使用图形化界面和虚拟机交互，当有图形化framebuffer的时候，输入设备会被自动提供的。

<input type='mouse' bus='ps2'/>

input元素：input元素含有一个强制的属性，type属性的值可以是mouse活tablet，前者使用想对运动，后者使用绝对运动。bus属性指定一个明确的设备类型，值可以是：xen、ps2、usb。

图形设备：

图形设备允许有个图形接口和客户端进行交互，客户端有图形接口和text console模式允许admin进行交互。

<graphics type='vnc' port='-1' keymap='en-us'/>

<graphics type='vnc' port='5904'>

<listen type='address' address='1.2.3.4'/>

</graphics>

graphics元素：graphics含有一个强制的属性type，type的值可以是：sdl、vnc、rdp、desktop。vnc则启动vnc服务，port属性指定tcp端口，如果是-1，则表示自动分配，vnc的端口自动分配的话是从5900向上递增。listen属性提供一个IP地址给服务器监听，可以单独在listen元素中设置。passwd属性提供一个vnc的密码。keymap属性提供一个keymap使用。

Rather than putting the address information used to set up the listening socket for graphics types vnc and spice in the <graphics> listen attribute, a separate subelement of <graphics>, called <listen> can be specified (see the examples above)since 0.9.4. <listen> accepts the following attributes:

listen元素：listen元素专门针对vnc和spice设置监听端口等。它包含以下属性：type、address、network。type的值可以是address或network。如果设置了type=address，那么address属性设置一个ip地址或者主机名来监听。如果type=network，则network属性设置一个网络名称在libvirt‘s的网络配置文件中。

字符设备提供同虚拟机进行交互的接口，Paravirtualized consoles, serial ports, parallel ports and channels 都是字符设备，它们使用相同的语法。

串行端口：

<serial type='pty'>

<target port='0'/>

</serial>

<console type='pty'>

<target type='serial' port='0'/>

</console>

Pseudo TTY 分配使用/dev/ptmx，A suitable client such as 'virsh console' can connect to interact with the serial port locally.

<parallel type='pty'>

<source path='/dev/pts/2'/>

<target port='0'/>

</parallel>

在每组指令中，最顶层的指令(parallel, serial, console, channel)描述设备怎样出现在客户端中，客户端接口通过target配置。

The interface presented to the host is given in the type attribute of the top-level element. The host interface is configured by the source element

主机接口通过source元素配置。

声音设备：

<video>

<model type='cirrus'/>

</video>

video元素：是描述声音设备的容器，为了向后完全兼容，如果没有设置video但是有graphics在xml配置文件中，这时libvirt会按照客户端类型增加一个默认的video，。model元素有一个强制的type属性，它的值可以是：vga、cirrus、vmvga、xen、vbox、qxl。例如一个客户端类型为kvm，那么默认的type值是cirrus。

</devices>

**下面kvm配置文件几个修改实例：**

Kvm配置文件xml修改常用调整的主要有cpu、内存、网卡、磁盘、启动级别、vnc端口

1、Kvm虚拟机内存调整修改如下配置中内存值，内存大小为1024\*1024\*n，n是以G为单位：

<memory unit='KiB'>1048576</memory>

<currentMemory unit='KiB'>1048576</currentMemory>

2、kvmcpu核数调整，下面是1核

<vcpu placement='static'>1</vcpu>

3、如果需要救援模式，需要修改配置文件xml里面的启动，光驱启动

编辑配置文件找到<disk type='block' device='cdrom'>

将下面修改

<disk type='file' device='cdrom'>

增加源

<source file='/var/system\_iso/CentOS-6.5-x86\_64-bin-DVD1.iso'/>

如果需要救援模式需要修改第一个disk里面的

<boot dev='hd'/> 改为<boot dev='cdrom'/>救援模式结束以后记得改回来硬盘启动

4、修改vnc端口，kvm虚拟机中每个子机的vnc端口都不一样

<graphics type='vnc' port='5902' autoport='no' listen='0.0.0.0'>

**九、Kvm中常见问题与解决：**

**9.1、在kvm主机中执行Virsh shutdown kvm1 无法关闭guest**

解决办法由于关机通过acpi电源管理接口来实现的登录kvm1虚拟机执行yum install acpid 开机acpid以后就可以正常关机

**9.2、解决virsh console kvm无法连接到guset的问题**

下面的操作必须在guest虚拟机上面执行，登录到guest虚拟机执行下面的操作

1、修改 /etc/securetty 到最后添加 ttyS0

echo ttyS0 >> /etc/securetty

2、修改/etc/inittab 到最后添加 S0:12345:respawn:/sbin/agetty/ ttyS0 115200

echo "S0:12345:respawn:/sbin/agetty/ ttyS0 115200" >> /etc/inittab

1. 修改/etc/grub.conf ，如下图所示在 kernel 这一行的末尾，添加console=ttyS0
2. 重启reboot，再在主机使用virsh console kvm1 连接ok

**9.3、在kvm主机中无法使用virt-viewer连接。**

执行virt-viewer -c qemu:///system test1报错如下

Cannot open display:

Run 'virt-viewer --help' to see a full list of available command line options

解决办法需要修改一下display

export DISPLAY=127.0.0.1:10.0不然连接不上图形界面