# 第1章

# 部署 KVM 虚拟化平台

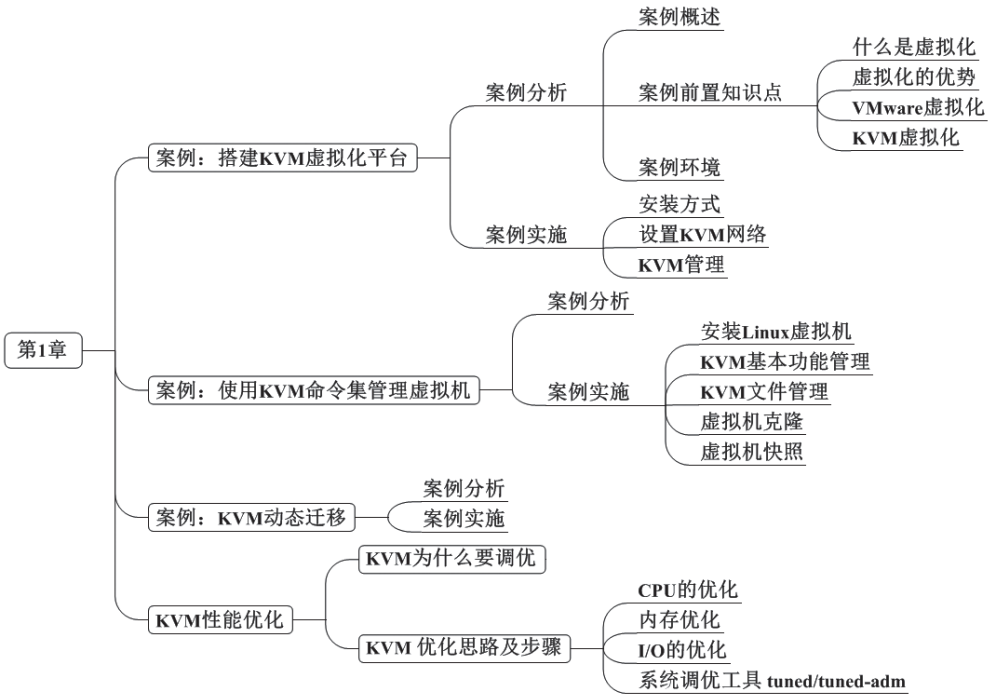
## 技能目标

- 理解 KVM 架构
- 会部署虚拟化环境
- 会创建虚拟机实例
- 会进行 KVM 动态迁移
- 会优化 KVM 性能

## 本章导读

KVM 是 Kernel Virtual Machine 的简写，目前 Red Hat 只支持在 64 位的 RHEL5.4 及以上的系统运行 KVM，同时硬件需要支持 VT 技术。KVM 的前身是 QEMU，2008 年被 Red Hat 公司收购并获得一项 hypervisor 技术，不过 Red Hat 的 KVM 被认为将成为未来 Linux hypervisor 的主流。准确来说，KVM 仅仅是 Linux 内核的一个模块。管理和创建完整的 KVM 虚拟机，需要更多的辅助工具。本章将介绍部署虚拟化环境、创建虚拟机实例，以及虚拟机的基本管理。

## 知识服务






## 1.1 案例：搭建 KVM 虚拟化平台

### 1.1.1 案例分析

#### 1. 案例概述

公司现有部分 Linux 服务器利用率不高，为充分利用这些 Linux 服务器，可以部署 KVM，在物理机上运行多个业务系统。例如，在运行 Nginx 的服务器上部署 KVM，然后在虚拟机上运行 Tomcat。

#### 2. 案例前置知识点

（1）什么是虚拟化

虚拟化就是把硬件资源从物理方式转变为逻辑方式，打破原有物理结构，使用户可以灵活管理这些资源，并且允许 1 台物理机上同时运行多个操作系统，以实现资源利用率最大化和灵活管理的一项技术。

（2）虚拟化的优势

1）减少服务器数量，降低硬件采购成本。

2）资源利用率最大化。

3）降低机房空间、散热、用电消耗的成本。

4）硬件资源可动态调整，提高企业 IT 业务灵活性。

5）高可用性。

6）在不中断服务的情况下进行物理硬件调整。

7）降低管理成本。

8）具备更高效的灾备能力。

（3）VMware 虚拟化

vSphere 是 VMware 公司在 2001 年基于云计算推出的一套企业级虚拟化解决方案，核心组件为 ESX，现在已经被 ESXi 取代。该产品经历 5 个版本的改进，已经实现了虚拟化基础架构、高可用性、集中管理、性能监控等一体化的解决方案，目前仍在不断扩展增强，功能越来越丰富，号称是业界第一套云计算的操作系统。

ESXi 是 VMware 服务器虚拟化体系的重要成员之一，也是 VMware 服务器虚拟化的基础。其实它本身也是一个操作系统，采用 Linux 内核（VMKernel），安装方式为裸金属方式，直接安装在物理服务器上，不需要安装任何其他操作系统。为了使它尽可能小地占用系统资源，同时又可以保证其高效稳定的运行，VMware 将其进行了精简封装。

（4）KVM 虚拟化

KVM 自 Linux 2.6.20 版本后就直接整合到 Linux 内核中，它依托 CPU 虚拟化指令集（如 Intel-VT、AMD-V）实现高性能的虚拟化支持。由于与 Linux 内核高度整合，因此在性能、安全性、兼容性、稳定性上都有很好的表现。

图 1.1 简单描绘了 KVM 虚拟化架构，在 KVM 环境中运行的每一个虚拟化操作系统都将表现为单个独立的系统进程。因此它可以很方便地与 Linux 系统中的安全模块进行整合（SELinux），可以灵活地实现资源的管理及分配。

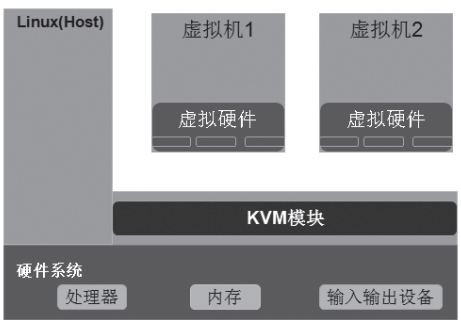



图 1.1　KVM 虚拟化架构

### 3. 案例环境

采用 CentOS 6.5 x86_64，开启 CPU 虚拟化支持。

## 1.1.2　案例实施

### 1. 安装方式

（1）最简单的安装方法就是在安装系统的时候，选择桌面安装，然后选择“虚

拟化”选项，如图 1.2 和图 1.3 所示。


图 1.2　桌面安装


图 1.3　安装虚拟化平台

（2）在已有系统基础上，安装 KVM 所需软件。

```
yum -y groupinstall "Desktop"              // 安装 GNOME 桌面环境
yum -y install qemu-kvm.x86_64             // KVM 模块
yum -y install qemu-kvm-tools.x86_64       // KVM 调试工具，可不安装
yum -y install python-virtinst.noarch      // python 组件，记录创建 VM 时的 xml 文件
yum -y install qemu-img.x86_64             // qemu 组件，创建磁盘、启动虚拟机等
yum -y install bridge-utils.x86_64         // 网络支持工具
yum -y install libvirt                     // 虚拟机管理工具
yum -y install virt-manager                // 图形界面管理虚拟机
```

（3）验证。重启系统后，查看 CPU 是否支持虚拟化，对于 Intel 的服务器可以通过以下命令查看，只要有输出就说明 CPU 支持虚拟化；对于 AMD 的服务器可以用 cat/proc/cpuinfo | grep smv 命令查看。

```
[root@kgc ~]# cat /proc/cpuinfo | grep vmx
flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
       pse36 clflush dts mmx fxsr sse sse2 ss syscall nx rdtscp lm constant_tsc up
       arch_perfmon pebs bts xtopology tsc_reliable nonstop_tsc aperfmperf unfair_
       spinlock pni pclmulqdq vmx ssse3 cx16 pcid sse4_1 sse4_2 x2apic popcnt xsave
       avx hypervisor lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi ept
       vpid fsgsbase smep
```

检查 KVM 模块是否安装：

```
[root@kgc ~]# lsmod | grep kvm
kvm_intel             54285  0
kvm                  333172  1 kvm_intel
```



### 2. 设置 KVM 网络

宿主服务器安装完成 KVM，首先要设定网络，在 libvirt 中运行 KVM 网络有两种模式：NAT 和 Bridge，默认是 NAT。

关于两种模式的说明：

（1）用户模式，即 NAT 模式，这种模式是默认网络，数据包由 NAT 模式通过主机的接口进行传送，可以访问外网，但是无法从外部访问虚拟机网络。

（2）桥接模式，这种模式允许虚拟机像一台独立的主机那样拥有网络，外部的机器可以直接访问到虚拟机内部，但需要网卡支持，一般有线网卡都支持。

这里以 Bridge（桥接）模式为例。

```
[root@kgc ~]#vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=none
NM_CONTROLLED=no
ONBOOT=yes
TYPE=Ethernet
HWADDR=00:0c:29:0c:6b:48
BRIDGE="br0"

[root@kgc ~]#vi /etc/sysconfig/network-scripts/ifcfg-br0
DEVICE=br0
BOOTPROTO=static
NM_CONTROLLED=no
ONBOOT=yes
TYPE=Bridge
IPADDR=192.168.10.1
NETMASK=255.255.255.0
```

重启 network 服务：

```
[root@kgc ~]# service  network restart
正在关闭接口 br0：                          [ 确定 ]
正在关闭接口 eth0：                         [ 确定 ]
关闭环回接口：                              [ 确定 ]
弹出环回接口：                              [ 确定 ]
弹出界面 eth0：                             [ 确定 ]
弹出界面 br0：  Determining if ip address 192.168.10.1 is already in use for device br0……
                                            [ 确定 ]
```

确认 IP 地址信息：

```
[root@kgc ~]# ifconfig
br0     Link encap:Ethernet  HWaddr 00:0C:29:0C:6B:48
        inet addr:192.168.10.1  Bcast:192.168.10.255  Mask:255.255.255.0
        inet6 addr: fe80::20c:29ff:fe0c:6b48/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:3034 errors:0 dropped:0 overruns:0 frame:0
        TX packets:331 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:235542 (230.0 KiB)  TX bytes:47392 (46.2 KiB)

eth0    Link encap:Ethernet  HWaddr 00:0C:29:0C:6B:48
        inet6 addr: fe80::20c:29ff:fe0c:6b48/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:4976 errors:0 dropped:0 overruns:0 frame:0
        TX packets:365 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:547785 (534.9 KiB)  TX bytes:50782 (49.5 KiB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:56 errors:0 dropped:0 overruns:0 frame:0
        TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:3992 (3.8 KiB)  TX bytes:3992 (3.8 KiB)

virbr0  Link encap:Ethernet  HWaddr 52:54:00:B0:82:E3
        inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

出现以上信息，就说明网卡桥接成功了。

### 3. KVM 管理

```
[root@kgc ~]#virt-manager
```

virt-manager 是基于 libvirt 的图形化虚拟机管理软件。请注意，不同发行版上 virt-manager 的版本可能不同，图形界面和操作方法也可能不同。本文使用了 CentOS 6 企业版。创建 KVM 虚拟机最简单的方法是通过 virt-manager 接口。从控制台窗口启动这个工具，以 root 身份输入 virt-manager 命令，出现如图 1.4 所示窗口。


图 1.4　虚拟机管理界面

虚拟化管理步骤如下。

（1）创建存储池，双击 localhost(QEMU), 选择“存储”选项卡，然后单击“+”按钮新建存储池。如图 1.5 所示，建立一个镜像存储池，命名为 bdqn。单击“前进”按钮，根据提示输入或浏览用以设置存储目录，如 /data_kvm/store，最后单击“完成”按钮即可。


图 1.5　创建存储池

（2）以同样的操作创建一个镜像存储池，命名为 bdqn_iso，目录为 /data_kvm/iso 即可。在安装操作系统时，我们把镜像上传到服务器目录 /data_kvm/iso，如图 1.6 所示。

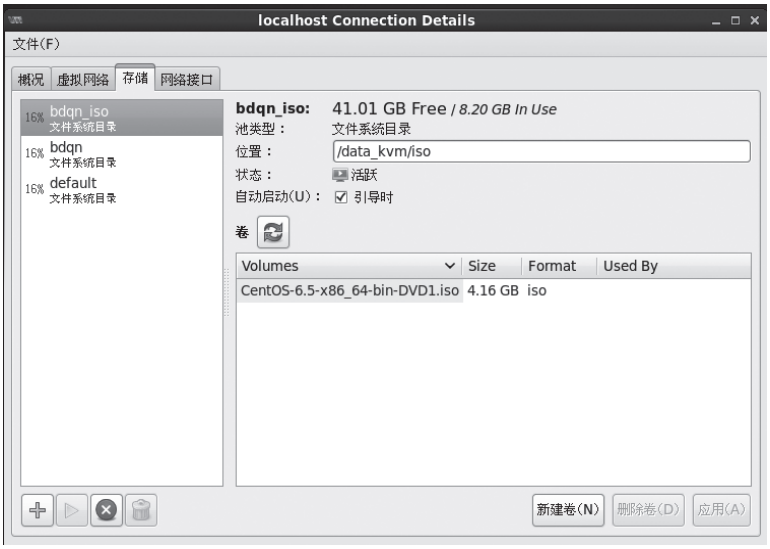



图 1.6　创建镜像存储池

（3）创建存储卷，单击刚创建好的“bdqn”，单击对话框右下角的“新建卷”按钮建立一个存储卷，并设置最大容量与分配容量，如图 1.7 所示。


图 1.7　创建存储卷

（4）单击“完成”按钮后，回到虚拟系统管理器。右击“Localhost(QEMU)”，然后选择“新建”选项，在弹出的对话框中按图 1.8 所示将虚拟机命名为“CentOS6.5”，然后单击“前进”按钮。

单击“浏览”按钮选择镜像文件，再选择操作系统类型及版本，如图 1.9 所示。

单击“前进”按钮，在图 1.10 所示的对话框中适当分配内存和 CPU 资源，如 1 核 CPU、1024MB 内存。


图 1.8　新建虚拟机（1）


图 1.9　新建虚拟机（2）

单击“前进”按钮，在如图 1.11 所示的对话框中勾选“立即分配整个磁盘”复选框，选择“选择管理的或者其他现有存储”单选按钮，单击“浏览”按钮选择文件，然后单击“前进”按钮。


图 1.10　新建虚拟机（3）


图 1.11　新建虚拟机（4）

在如图 1.12 所示的对话框中勾选“在安装前自定义配置”复选框，单击“完成”按钮，弹出如图 1.13 所示的对话框。

在“Overview”视图中，定位到“机器设置”，把“机器设置 - 时钟偏移”改为“localtime”，单击“应用”按钮即可。定位到“Boot Options”，勾选“主机引导时启动虚拟机”复选框，这样在物理宿主机启动后，这个 VM 也会启动，最后单击“应用”按钮，如图 1.14 所示。如果要远程管理，需要在“显示 VNC”中，将 Keymap 设置为“Copy Local Keymap”。


图 1.12　新建虚拟机（5）




图 1.13　新建虚拟机（6）

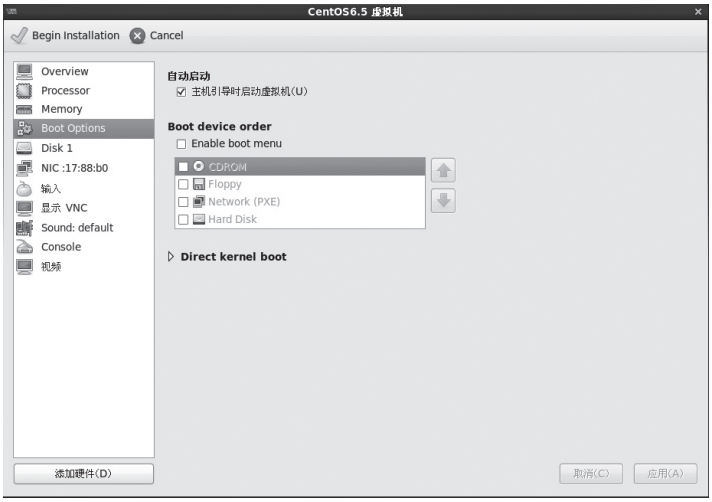



图 1.14　新建虚拟机（7）

最后单击“Begin Installation”按钮即可，整个虚拟化配置过程完成。下面就是安装操作系统的工作，和平时安装 Linux 系统一样，如图 1.15 所示。


图 1.15　CentOS 安装界面

## 1.2　案例：使用 KVM 命令集管理虚拟机

### 1.2.1　案例分析

案例环境使用一台物理机器（见表 1-1），一台服务器安装 CentOS 6.5 的 64 位系统（即 kgc），test01 是在宿主机 kgc 中安装的虚拟机。

表 1-1　KVM 虚拟化

| 主机 | 操作系统 | IP 地址 | 主要软件 |
|---|---|---|---|
| kgc | CentOS 6.5 x86_64 | 192.168.10.1 | Xshell<br>Xmanager |
| test01 | CentOS 6.5 x86_64 | 192.168.10.2 | |

### 1.2.2　案例实施

#### 1. 安装 Linux 虚拟机

安装过程同上一案例，使用 Xshell 远程控制 kgc 主机。

## 2. KVM 基本功能管理

（1）查看命令帮助

```
[root@kgc ~]# virsh -h
…… // 省略输出内容
```

（2）查看 KVM 的配置文件存放目录（test01.xml 是虚拟机系统实例的配置文件）

```
[root@kgc ~]# ls /etc/libvirt/qemu/
autostart networks  test01.xml
```

（3）查看虚拟机状态

```
[root@kgc ~]# virsh list --all
 Id 名称                状态
----------------------------------
 7 test01               running
```



（4）虚拟机关机与开机

首先需要确认 acpid 服务安装并运行。

```
[root@kgc ~]# virsh shutdown test01
[root@kgc ~]# virsh start test01
```

（5）强制实例系统关闭电源

```
[root@kgc ~]# virsh destroy test01
```

（6）通过配置文件启动虚拟机系统实例

```
[root@kgc ~]# virsh create /etc/libvirt/qemu/test01.xml
[root@kgc ~]# virsh list --all
 Id 名称                状态
----------------------------------
 10 test01              running
```

（7）挂起虚拟机

```
[root@kgc ~]# virsh suspend test01
```

查看虚拟机状态：

```
[root@kgc ~]# virsh list --all
 Id 名称                状态
----------------------------------
 10 test01              暂停
```

（8）恢复虚拟机

```
[root@kgc ~]# virsh resume test01

[root@kgc ~]# virsh list --all
 Id 名称                状态
```

```
----------------------------------
 10 test01              running
```

（9）配置虚拟机实例伴随宿主机自动启动

```
[root@kgc ~]# virsh autostart test01
```

上述命令将创建 /etc/libvirt/qemu/autostart/ 目录，目录内容为开机自动启动的系统。

（10）导出虚拟机配置

```
[root@kgc ~]# virsh dumpxml test01 > /etc/libvirt/qemu/test02.xml
```

（11）虚拟机的删除与添加

删除虚拟机：

```
[root@kgc ~]# virsh shutdown test01
[root@kgc ~]# virsh undefine  test01
```

查看删除结果，test01 的配置文件被删除，但是磁盘文件不会被删除。

```
[root@kgc ~]# ls /etc/libvirt/qemu/
autostart  networks  test02.xml
```

通过 virsh list --all 查看不到 test01 的信息，说明此虚拟机被删除。

```
[root@kgc ~]# virsh list --all
 Id 名称                状态
----------------------------------
```

通过备份的配置文件重新定义虚拟机：

```
[root@kgc ~]# cd  /etc/libvirt/qemu
[root@kgc qemu]# mv test02.xml test01.xml
```

重新定义虚拟机：

```
[root@kgc qemu]# virsh define test01.xml
```

查看虚拟机信息：

```
[root@kgc qemu]# virsh list --all
 Id 名称                状态
----------------------------------
 - test01               关闭
```

（12）修改虚拟机配置信息（用来修改系统内存大小、磁盘文件等信息）

直接通过 vim 命令修改：

```
[root@kgc ~]# vim /etc/libvirt/qemu/test01.xml
```

通过 virsh 命令修改：

```
[root@kgc ~]# virsh edit test01
```

### 3. KVM 文件管理

通过文件管理可以直接查看、修改、复制虚拟机的内部文件。例如，当系统因为配置问题无法启动时，可以直接修改虚拟机的文件。虚拟机磁盘文件有 raw 与 qcow2 两种格式，KVM 虚拟机默认使用 raw 格式，raw 格式性能最好、速度最快，其缺点是不支持一些新的功能，如镜像、Zlib 磁盘压缩、AES 加密等。针对两种格式的文件有不同的工具可供选择。这里介绍本地 YUM 安装 libguestfs-tools 后产生的命令行工具（这个工具可以直接读取 qcow2 格式的磁盘文件，因此需要将 raw 格式的磁盘文件转换成 qcow2 的格式）。

（1）转换 raw 格式磁盘文件至 qcow2 格式。

查看当前磁盘格式：

```
[root@kgc ~]# qemu-img info /data_kvm/store/test01.img
image: /data_kvm/store/test01.img
file format: raw
virtual size: 10G (10737418240 bytes)
disk size: 10G
```

关闭虚拟机：

```
[root@kgc ~]# virsh shutdown test01
```

转换磁盘文件格式：

```
[root@kgc ~]# qemu-img convert -f raw -O qcow2 /data_kvm/store/test01.img
      /data_kvm/store/test01.qcow2
```

（2）修改 test01 的 xml 配置文件。

```
[root@kgc ~]# virsh edit test01
…… // 省略部分内容
  <disk type='file' device='disk'>
    <driver name='qemu' type='qcow2' cache='none'/>
    <source file='/data_KVM/store/test01.qcow2'/>
    <target dev='vda' bus='virtio'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
  </disk>
…… // 省略部分内容
```

（3）virt-cat 命令类似于 cat 命令。

```
[root@kgc ~]# virt-cat -a /data_KVM/store/test01.qcow2 /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=test01
```

（4）virt-edit 命令用于编辑文件，用法与 vim 基本一致。

```
[root@kgc ~]# virt-edit  -a /data_KVM/store/test01.qcow2 /etc/resolv.conf
nameserver 8.8.8.8
```

（5）virt-df 命令用于查看虚拟机磁盘信息。

```
[root@kgc ~]# virt-df -h test01
Filesystem                      Size      Used  Available  Use%
test01:/dev/sda1                 484M       32M       427M    7%
test01:/dev/VolGroup/lv_root      7.4G      1.6G       5.4G   22%
```

### 4. 虚拟机克隆

（1）查看虚拟机状态

```
[root@kgc ~]# virsh list --all
 Id 名称                状态
----------------------------------
 - test01               关闭
```

（2）从 test01 克隆 test02

```
[root@kgc ~]# virt-clone -o test01 -n test02 -f /data_kvm/store/test02.qcow2
```

（3）查看虚拟机状态

```
[root@kgc ~]# virsh list --all
 Id 名称                状态
----------------------------------
 - test01               关闭
 - test02               关闭
```

（4）启动虚拟机

```
[root@kgc ~]# virsh start test02
```

### 5. 虚拟机快照

KVM 虚拟机要使用镜像功能，磁盘格式必须为 qcow2，之前已经将 test01 的磁盘格式转换成了 qcow2。

下面介绍 KVM 虚拟机快照备份的过程。

（1）对 test01 创建快照

```
[root@kgc ~]# virsh snapshot-create  test01
Domain snapshot 1382572463 created
```

（2）查看虚拟机快照版本信息

```
[root@kgc ~]# virsh snapshot-current test01
<domainsnapshot>
  <name>1382572463</name>                    // 快照版本号
  <state>running</state>
……                                          // 省略部分输出
```

（3）查看快照信息

```
[root@kgc ~]# virsh snapshot-list  test01
 名称                 Creation Time             状态
------------------------------------------------------------
 1382572463           2013-10-24 07:54:23 +0800 running
```

（4）创建新快照

```
[root@kgc ~]# virsh snapshot-create  test01
Domain snapshot 1382572597 created
```

（5）查看快照信息

```
[root@kgc ~]# virsh snapshot-list  test01
 名称                 Creation Time             状态
------------------------------------------------------------
 1382572463           2013-10-24 07:54:23 +0800 running
 1382572597           2013-10-24 07:56:37 +0800 running
```



（6）恢复虚拟机状态至 1382572463

```
[root@kgc ~]# virsh snapshot-revert test01 1382572463
```

（7）查看虚拟机快照版本信息

```
[root@kgc ~]# virsh snapshot-current test01
<domainsnapshot>
  <name>1382572463</name>                    // 快照版本号
  <state>running</state>
……                                           // 省略部分输出
```

（8）删除快照

```
[root@kgc ~]# virsh snapshot-delete test01 1382572463
Domain snapshot 1382572463 deleted
```

## 1.3 案例：KVM 动态迁移

### 1.3.1 案例分析

迁移是指将在 KVM 上运行的虚拟机系统转移到其他物理机的 KVM 上运行，可分为静态迁移和动态迁移。静态迁移是在虚拟机关机的情况下迁移，而动态迁移是在虚拟机上服务正常运行的情况下迁移，要基于共享存储。动态迁移仅有非常短暂的停机时间，不会对最终用户造成明显影响。

案例拓扑图如图 1.16 所示。

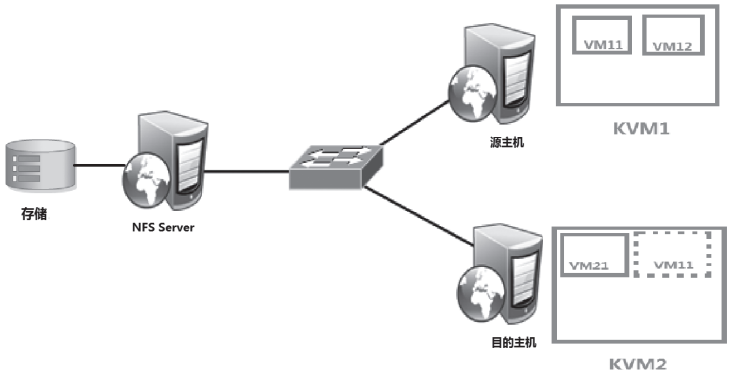



图 1.16　KVM 迁移拓扑图



### 1.3.2　案例实施

案例实施步骤如下：

（1）设置主机名、/etc/hosts，保证网络连接。

（2）两台主机的 KVM 连接 NFS 共享存储。

（3）在源主机的 KVM 中新建虚拟机并安装系统。

（4）连接 KVM，并进行迁移。

详细的操作请上课工场 APP 或官网 kgc.cn 观看视频。

## 1.4　KVM 性能优化

#### 1. KVM 为什么要调优

性能的损耗是关键。KVM 采用全虚拟化技术，全虚拟化要由一个软件来模拟硬件层，故有一定的损耗，特别是 I/O，因此需要优化。

KVM 性能优化主要在 CPU、内存、I/O 这几方面。当然对于这几方面的优化，也是要分场景的，不同的场景其优化方向也是不同的。

#### 2. KVM 优化思路及步骤

KVM 的性能已经很不错了，但还有一些微调措施可以进一步提高 KVM 性能。

（1）CPU 优化

要考虑 CPU 的数量问题，所有 guestcpu 的总数目不要超过物理机 CPU 的总数目。如果超过，则将对性能带来严重影响，建议选择复制主机 CPU 配置，如图 1.17 所示。

（2）内存优化

1）KSM（Kernel Samepage Merging，相同页合并）

内存分配的最小单位是 page（页面），默认大小是 4KB。可以将 host 机内容相同

的内存合并，以节省内存的使用，特别是在虚拟机操作系统都一样的情况下，肯定会有很多内容相同的内存值，开启了 KSM，则会将这些内存合并为一个，当然这个过程会有性能损耗，所以开启与否，需要考虑使用场景。

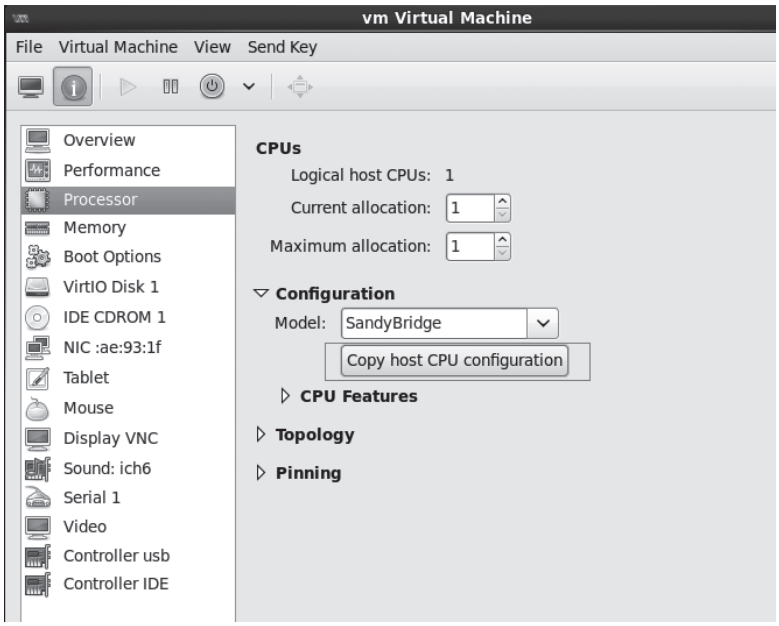



图 1.17　CPU 优化

而 KSM 对 KVM 环境有很重要的意义，当 KVM 上运行许多相同系统的客户机时，客户机之间将有许多内存页是完全相同的，特别是只读的内核代码页完全可以在客户机之间共享，从而减少客户机占用的内存资源，能同时运行更多的客户机。

通过 /sys/kernel/mm/ksm 目录下的文件可查看内存页共享的情况：

```
[root@localhostksm]# pwd
/sys/kernel/mm/ksm
[root@localhostksm]# ll
total 0
-r--r--r--. 1 root root 4096 Jan 24 19:15 full_scans
-rw-r--r--. 1 root root 4096 Jan 24 19:15 merge_across_nodes
-r--r--r--. 1 root root 4096 Jan 24 19:15 pages_shared
-r--r--r--. 1 root root 4096 Jan 24 19:15 pages_sharing
-rw-r--r--. 1 root root 4096 Jan 24 19:15 pages_to_scan
-r--r--r--. 1 root root 4096 Jan 24 19:15 pages_unshared
-r--r--r--. 1 root root 4096 Jan 24 19:15 pages_volatile
-rw-r--r--. 1 root root 4096 Jan 24 19:15 run
-rw-r--r--. 1 root root 4096 Jan 24 19:15 sleep_millisecs
```

pages_shared 文件中记录了 KSM 共享的总页面数；

pages_sharing 文件中记录了当前共享的页面数。

每个页面的大小为 4KB，可计算出共享内存为：4× 页面数 = 内存大小（KB）。

```
[root@localhostksm]# cat run   // 是否开启 KSM，0 是不开启，1 是开启
0
[root@localhostksm]# echo 1 >run        // 临时开启 KSM，只能使用重定向，不支持用 VI 编辑器，
                                        // 可以在 /etc/rc.local 中添加 echo 1 > /sys/kernel/mm/ksm/
                                        // run 让 KSM 开机自动运行
[root@localhostksm]# cat run
1
[root@localhostksm]# cat pages_to_scan    // 定期扫描相同页，sleep_millisecs 决定多长时间，
                                          // pages_to_scan 决定每次查看多少个页面，
100 // 默认为 100，越大越好，超过 2000 无效，需要开启两个服务 ksmtuned 和 tuned
    // 支持更多页面
```

KSM 会稍微影响系统性能，以效率换空间，如果系统的内存很宽裕，则无须开启 KSM，如果想尽可能多地并行运行 KVM 客户机，则可以打开 KSM。

2）对内存设置限制

如果我们有多个虚拟机，为了防止某个虚拟机无节制地使用内存资源，导致其他虚拟机无法正常使用，就需要对内存的使用进行限制。

```
[root@localhost ~]#virsh memtune vm        // 查看当前虚拟机 vm 内存的限制，单位为 KB
Hard_limit: unlimited                      // 强制最大内存
Soft_limit: unlimited                      // 可用最大内存
Swap_hard_limit: unlimited                 // 强制最大 swap 使用大小
```

语法：

```
[root@localhost ~]# virsh memtune --help
......
OPTIONS
   [--domain] <string>  domain name, id or uuid
   --hard-limit <number>  Max memory, as scaled integer (default KiB)
   --soft-limit <number>  Memory during contention, as scaled integer (default KiB)
   --swap-hard-limit <number>  Max memory plus swap, as scaled integer (default KiB)
   --min-guarantee <number>  Min guaranteed memory, as scaled integer (default KiB)
                                          // 保证最小内存
   --config        affect next boot       // 下次重启生效
   --live          affect running domain  // 在线生效
   --current       affect current domain  // 只在当前生效
```

例如：

```
[root@localhost ~]# virsh memtune vm  --hard-limit 1024000 --live   // 设置强制最大内存 100MB，
                                                                    // 在线生效
```

3）大页后端内存（Huge Page Backed Memory）

在逻辑地址向物理地址转换时，CPU 保持一个翻译后备缓冲器 TLB，用来缓存转换结果，而 TLB 容量很小，所以如果 page 很小，TLB 很容易就充满，这样就容易导致 cache miss，相反 page 变大，TLB 需要保存的缓存项就变少，就会减少 cache

miss。通过为客户机提供大页后端内存，就能减少客户机消耗的内存并提高TLB命中率，从而提升 KVM 性能。

Intel 的 x86 CPU 通常使用 4KB 内存页，但是经过配置，也能够使用大页（huge page）：x86_32 是 4MB，x86_64 和 x86_32 PAE 是 2MB，这是 KVM 虚拟机的又一项优化技术。

使用大页，KVM 的虚拟机的页表将使用更少的内存，并且将提高 CPU 的效率。

```
[root@localhost ~]#cat /proc/meminfo // 查看内存信息，无可用大页
AnonHugePages:    346112 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0

[root@localhost ~]#echo 25000>/proc/sys/vm/nr_hugepages      // 指定大页需要的内存页面数
                                                             //（临时生效）
[root@localhost ~]#cat /proc/meminfo |grep HugePage
AnonHugePages:    100352 kB
HugePages_Total:     999
HugePages_Free:      999     // 现在没有任何软件在使用大页
HugePages_Rsvd:        0
HugePages_Surp:        0

[root@localhost ~]#sysctl -w vm.nr_hugepages=25000// 指定大页需要的内存页面数永久生效
```

或者在 /etc/sysctl.conf 中添加 vm.nr_hugepages=2500 来持久设定大页文件系统需要的内存页面数。

**注意**

大页文件系统需要的页面数可以由客户机需要的内存除以页面大小来大体估算。

```
[root@localhost ~]#virsh destroy vm      // 关闭虚拟机 vm
Domain vm destroyed

[root@localhost ~]#virsh edit vm         // 编辑虚拟机的 XML 配置文件使用大页来分配内存
<domain type='kvm'>
<name>vm</name>
<uuid>ba39310e-f751-5649-ac93-845c5d339b84</uuid>
<memory unit='KiB'>1048576</memory>
<currentMemory unit='KiB'>1048576</currentMemory>
<memoryBacking><hugepages/></memoryBacking>    // 添加，使用大页
```

保存退出。

```
[root@localhost ~]#mount -t hugetlbfshugetlbfs /dev/hugepages// 挂载 hugetlbfs 文件系统

[root@localhost ~]#service libvirtd restart
[root@localhost ~]#virsh start vm
Domain vm started

[root@localhost ~]#cat /proc/meminfo |grep HugePage
AnonHugePages:     59392 kB
HugePages_Total:    999
HugePages_Free:     873
HugePages_Rsvd:     394
HugePages_Surp:       0

[root@localhost ~]#virsh destroy vm
Domain vm destroyed

 [root@localhost ~]#cat /proc/meminfo |grep HugePage   // 被释放
AnonHugePages:     26624 kB
HugePages_Total:    999
HugePages_Free:     999
HugePages_Rsvd:       0
HugePages_Surp:       0
```

让系统开机自动挂载 hugetlbfs 文件系统，在 /etc/fstab 中添加：

```
hugetlbfs/hugepages hugetlbfs defaults 0 0
```

（3）I/O 的优化

在实际的生产环境中，为了避免过度消耗磁盘资源而对其他的虚拟机造成影响，我们希望每台虚拟机对磁盘资源的消耗是可以控制的。比如多个虚拟机往硬盘中写数据，谁可以优先写，就可以调整 I/O 的权重 weight，权重越高写入磁盘的优先级越高。

对磁盘 I/O 控制有两种方式：

1）在整体中的权重，范围在 100 ～ 1000。

2）限制具体的 I/O。

```
[root@localhost ~]#virsh blkiotune vm
weight       : 0
device_weight  :

[root@localhost ~]#virsh blkiotune vm --weight 500  // 设置权重为 500
[root@localhost ~]#virsh blkiotune vm
weight       : 500
device_weight  :
```

编辑虚拟机的 XML 配置文件：

```
<blkiotune><weight>500</weight></blkiotune>
```

还可以使用 blkdeviotune 设置虚拟机的读写速度，语法如下：

```
[root@localhost ~]# virsh blkdeviotune --help
......
--total-bytes-sec <number>  total throughput limit in bytes per second
   --read-bytes-sec <number>  read throughput limit in bytes per second
   --write-bytes-sec <number>  write throughput limit in bytes per second
   --total-iops-sec <number>  total I/O operations limit per second
   --read-iops-sec <number>  read I/O operations limit per second
   --write-iops-sec <number>  write I/O operations limit per second
   --config        affect next boot
   --live          affect running domain
--current       affect current domain
```



（4）系统调优工具 tuned/tuned-adm

CentOS 在 6.3 版本以后引入了一套新的系统调优工具 tuned/tuned-adm，其中，tuned 是服务端程序，用来监控和收集系统各个组件的数据，并依据数据提供的信息动态调整系统设置，达到动态优化系统的目的；tuned-adm 是客户端程序，用来和 tuned 打交道，用命令行的方式管理和配置 tuned/tuned-adm，提供了一些预先配置的优化方案可供直接使用。当然不同的系统和应用场景有不同的优化方案，tuned-adm 预先配置的优化策略不是总能满足要求，这时候就需要定制，tuned-adm 允许用户自己创建和定制新的调优方案。

```
[root@localhost ~]# yum install tuned        // 安装和启动 tuned 工具
[root@localhost ~]# service tuned start
[root@localhost ~]# chkconfig tuned on
[root@localhost ~]# service ktune start
[root@localhost ~]# chkconfig ktune on
[root@localhost ~]# tuned-adm active         // 查看当前优化方案
Current active profile: default
Service tuned: enabled, running
Service ktune: enabled, running


[root@localhost ~]# tuned-adm list           // 查看预先设定好的优化方案
Available profiles:
- sap
- desktop-powersave
- virtual-guest          // 企业级服务器配置中使用这个 profile，其中包括电池备份控制程序、
                         // 缓存保护以及管理磁盘缓存
- spindown-disk
- latency-performance   // 延迟性能调试的服务器配置
- enterprise-storage    // 企业存储服务器优化方案
- laptop-ac-powersave
- default               // 默认节电配置，是最基本的节点配置，只启用磁盘和 CPU 插件
- laptop-battery-powersave
```

```
- virtual-host              // 根据 enterprise-storage 配置，virtual-host 还可减少可置换的虚拟内存，
                            // 并启用更多集合脏页写回。同时推荐在虚拟化主机中使用这个配置，
                            // 包括 KVM 和红帽企业版 Linux 虚拟化主机
- throughput-performance        // 吞吐性能调整的服务器 profile。如果系统没有企业级存储，
                                // 则建议使用这个 profile
- server-powersave
Current active profile: default      // 使用某种 profile

[root@localhost ~]# tuned-adm profile virtual-guest   // 修改优化方案为 virtual-guest
Stopping tuned:                                [  OK  ]
Switching to profile 'virtual-guest'
Applying deadline elevator: dm-0 dm-1 sda[  OK  ]
Applying ktunesysctl settings:
/etc/ktune.d/tunedadm.conf:                    [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh start':      [  OK  ]
Applying sysctl settings from /etc/sysctl.d/libvirtd
Applying sysctl settings from /etc/sysctl.conf
Starting tuned:                                [  OK  ]

[root@localhost ~]#tuned-adm active
Current active profile: virtual-guest
Service tuned: enabled, running
Service ktune: enabled, running
```

如果预定的方案不是总能满足要求，用户可以为自己的需求定制自己的优化方案。自己定制很容易，只需切换到优化方案的配置目录：/etc/tune-profiles/，不同的 profile 以目录的形式存在，拷贝其中一个例子，然后编辑里面的相关参数就可以了，使用 tuned-adm list 命令就能看到新创建的方案。

```
[root@localhost ~]# cd /etc/tune-profiles/
[root@localhost tune-profiles]# ls
active-profilelaptop-ac-powersavespindown-disk
default          laptop-battery-powersavethroughput-performance
desktop-powersave   latency-performance      virtual-guest
enterprise-storage sap                  virtual-host
functions          server-powersave
[root@localhost tune-profiles]# cp -r virtual-host my-server
```

编辑 my-server 目录中文件添加调优参数即可：

```
[root@localhost virtual-host]# tuned-adm list
Available profiles:
- my-server
- sap
- desktop-powersave
- virtual-guest
- spindown-disk
```

```
- latency-performance
- enterprise-storage
- laptop-ac-powersave
- default
- laptop-battery-powersave
- virtual-host
- throughput-performance
- server-powersave
Current active profile: virtual-guest
```

## 本章总结



- KVM 虚拟化平台作为 Linux 内核模块之一，依托 CPU 虚拟化指令集（如 Intel-VT、AMD-V）实现高性能的、安全稳定的虚拟化支持。
- KVM 对硬件的要求，必须是 64 位操作系统，其 CPU 支持 Intel 或 AMD 虚拟化，可以分别通过 cat /proc/cpuinfo | grep vmx 命令或 cat /proc/cupinfo | grep smv 命令查看。
- KVM 可通过命令集管理，但庞大的命令集很难清楚明了，所以通常都是在桌面环境下通过图形界面管理，直观方便。
- 迁移是指将在 KVM 上运行的虚拟机系统转移到其他物理机的 KVM 上运行，可分为静态迁移和动态迁移。
- KVM 是全虚拟化技术，全虚拟化由一个软件模拟硬件层，会有一定的损耗，特别是 I/O，因此我们需要进行优化。KVM 性能优化主要在 CPU、内存、I/O 这几个方面进行。

## 本章作业

1．自行搜索资料，深入理解 KVM 架构。

2．上课工场 APP 或官网 kgc.cn 观看 KVM 迁移视频，完成实验。