Xen 管理手册

Xen 管理手册

在前面的专题里，我们对 Xen 有了一定的了解，在本专题中，我们将继续学习如何管理 Xen。

块设备管理

在安装 Xen 虚拟机时，需要决定选用何种块设备（block devices）。本部分介绍几种最重要的选择。

- 虚拟机安装：Xen 块设备管理

硬件管理

本部分介绍 Xen 环境下的硬件管理，包括 PCI（外围部件互连）设备、准虚拟环境下的内存与 CPU 分配以及硬件驱动管理。

- 在虚拟化里管理 PCI 设备
- Xen 环境下的内存与 CPU 分配：性能优化
- Xen：完全虚拟化与准虚拟化的硬件驱动管理

创建快照

开源 Xen 不提供对快照的支持，而 Linux 能支持。由于开源 Xen 通常使用 Linux 作为它的特权域，所以你能使用 Linux 命令创建快照。那么有哪些命令可用呢？

- 使用 Linux 命令在 Xen 里创建快照

虚拟机安装：Xen 块设备管理

在安装 Xen 虚拟机时，需要决定选用何种块设备（block devices）。你有三种可能的选择：直接在专用分区中安装；使用磁盘镜像文件；使用 dd 创建磁盘镜像文件。在本文中，TechTarget 中国的特约虚拟化专家 Sander van Vugt 将概括地介绍最重要的几种选择。

使用物理磁盘

如果要获得最好的性能，物理磁盘设备是最好的选择。使用物理磁盘设备安装时，你可以安装到一个分区中或逻辑卷中（如果使用了逻辑卷），既可以使用 EVMS 也可以使用 LVM。使用逻辑卷有这样几个好处：

- 逻辑卷可以改变大小
- 可以使用快照进行备份

使用物理磁盘设备的最大好处是它提供的性能可能是最好的。然而，物理存储设备不便于移动。要拷贝磁盘镜像文件很容易，但要拷贝逻辑卷或物理分区就会困难得多。不过，困难得多并不表示不可能。例如，如果要将整个逻辑卷/dev/xen/vm1 拷贝到一个预建的逻辑卷/dev/xen/vm1-backup 中，只需使用 dd if=/dev/xen/vm1 of=/dev/xen/vm1-backup 即可。

要通过虚拟机配置文件处理物理磁盘设备，需要使用如下语句：

phy:,,[r|w]

例如：

disk=['phy:/dev/system/mainserver,xvda,w', 'phy:/dev/cdrom,xvdb,r',]

使用 Xen 磁盘镜像文件

在 SUSE 或 Red Hat 上安装虚拟机时，默认使用的是 Xen 磁盘镜像文件。该镜像文件是通过 dd 工具（dd utility）创建的，所以基本没有任何特殊之处。dd 磁盘镜像文件与虚拟硬盘（VHD）格式的文件不一样。dd 磁盘镜像文件没有相关的元数据（metadata），只有 0 和 1。

要创建磁盘镜像文件，你需要通过 dd 工具把/dev/zero 设备中的 0 复制到你的磁盘镜像文件中。例如，如果要创建一个 4GB 的文件（文件名为 /var/lib/xen/images/disk1），可以使用如下命令：

```
dd if=/dev/zero of=/var/lib/xen/images/disk1 bs=1M count=4096
```

除了使用 dd 工具创建空镜像文件以外，你还可以使用 ISO 文件替代物理光驱创建，用法和用 dd 一样。如果你甚至没有 ISO 文件，不要紧，ISO 文件很好创建。如果你要创建 ISO 文件的 cd-rom 在光驱中，使用如下命令将其复制到 cdrom.iso 文件：

```
dd if=/dev/cdrom of=/isos/cdrom.iso
```

创建好一个磁盘镜像文件或 ISO 文件后，你需要在虚拟机配置文件中包含该磁盘镜像文件，使用文件选项：

```
disk=[ 'file:/var/lib/xen/images/disk1,hda,w', 'file:/isos/cdrom.iso,hdc,r', ]
```

其它环境的镜像文件的使用

有时，你想从创建镜像文件开始创建一个新的虚拟机。而有些情况下，你要使用的磁盘镜像文件是来自另一种虚拟化环境，可能是 VMware 环境下的一个 vmdk-file，也可能是 Qemu 环境下的一个 qcow 文件或者其它文件。要使用这些文件，你必须 “tap” 磁盘镜像。如下命令行是使用 VMware 环境下的 vmdk-file 的语句：

```
disk=[ 'tap:vmdk:/var/lib/xen/server1.vmdk,hda,w', ]
```

Xen 磁盘镜像文件是一种特殊的 tap 设备，它使用的是 xen blktap 内核模块。这个模块允许在完全虚拟化环境下作为准虚拟设备处理磁盘镜像文件。要使用这个模块，你需要有 aio 类型的设备。磁盘镜像文件本身可能是一个由 dd 命令创建的普通磁盘镜像文件。要在配置文件中使用那种设备类型，可以使用如下语句：

```
disk=[ 'tap:aio:/var/lib/xen/server2.img,xvda,w', ]
```

使用 tap 磁盘镜像有个好处，就是可以使用任何类型的磁盘镜像。然而，它有个缺点：目前的管理工具并不支持 tap 设备，如 virtual manager。这就意味着你不得不利用这个设备手动配置虚拟机。

网络块设备

最后，还有一种在虚拟机中可以使用的磁盘类型是网络磁盘类型。它既可以是一个网络块设备（network block devices），如 DRBD 设备，也可以是 iSCSI 设备。这使得在 iSCSI SAN 中直接处理 LUN 成为可能，不过要记得先在 domain0 中启动 iSCSI 启动程序。

这个磁盘镜像类型是不可移动的，这就意味着你必须在 SAN 中做一些预防措施，以保护该类型的磁盘镜像文件。例如，在 SAN 级进行快照备份以保护这些文件。要连接一个 iSCSI LUN，你需要在 Xen 虚拟机的配置文件中包含完整的 iSCSI Qualifying Name（iqn）。例如，如果你要使用的 iqn 为 iqn.2007-08.nl.example:xendata，则需要用如下配置行：

```
disk=[ 'iscsi:2007-08.nl.example:xendata,xvda,w', ]
```

在本文中，我们了解到了处理虚拟机磁盘设备的各种不同方法。在多数情况下，即使使用了文件设备和物理设备，其它设备类型也是很有用的。尤其是 tap 设备类型，它能允许你处理任何类型的磁盘镜像文件。

（作者：Sander van Vugt 译者：涂凡才 来源：TechTarget 中国）

在虚拟化里管理 PCI 设备

在默认情况下，所有的 PCI（外围部件互连）设备能用于所有虚拟机。在某些情况下却不可用。例如，仅仅只能用于一台具体虚拟机的软件狗或网络板（network board），你想要把它们保留以达到更好的性能。在本文中，TechTarget 中国的特约虚拟化专家 Sander van Vugt 将分析与 PCI 设备相关的选项。本文使用的是 Xen 环境。

为一台虚拟机保留 PCI 设备，你必须确保不是由 Dom0 操作系统完成的。因此，你需要 pciback 模块。让这个模块自己工作，你必须确信它在每个启动程序的早期被激活。一般说来，你必须把它放在服务器的 initrd 里。如果你正使用 SUSE，打开 /etc/sysconfig/内核并添加 pciback 到 initrd 模块列表里。接下来，运行 mkinitrd 命令以产生新 initrd。

既然当你服务器启动时，你确定 pciback 模块首先装入，你能给它分配 PCI 接口。你需要用到表 1 所示的 lspci 命令。

（表 1 请参见文章末尾部分的“代码”部分）

在 lspci 命令输出的例子中，你看见一个简短的使用 PCI 设备 ID 的记法。在前面输入 0000 得到完整的设备 ID。接下来，使用/etc/modprobe.conf.local 配置文件告诉 pciback 应该排除某一个模块。例如，从上面的模块列表中包含 IEEE 1394，在 /etc/modprobe.conf 里输入下面行：

```
options pciback hide=(0000:03:01.0)
```

确保随后编辑加载模块配置文件时，能使用 mkinitrd 命令重建 initrd。同样不要忘记重新启动虚拟机。

用这种方法排除 PCI 设备后，你需要把它与某个虚拟机捆绑在一起。因此，你必须把它放进它的配置文件中，或当启动虚拟机时指定设备的名字。如果你想把它永久地放进虚拟机的配置文件里，在文件中输入下面行：

```
pci=[ '0000:03:01.0', ]
```

接下来，开启虚拟机。你现在能看见虚拟机里的 PCI 设备。这台虚拟机将是唯一能访问这个 PCI 设备的机器。你也能执行一个手动捆绑。例如，当启动虚拟机时，将下面的命令暂时分配给一个 PCI 设备：

```
xm create pci=0000:03:01.0 /etc/xen/vm/virtualserver
```

当在一个使用 Virtual Machine Manager 的虚拟化平台上作出这样的设置时，接下来不要忘记使用 xm new /etc/xen/vm/virtualserver 命令。这也将添加虚拟机到托管的 Xen 环境。

设备共享的好坏

在虚拟环境工作的主要好处是共享物理设备的能力。然而某些情况下，不要共享设备，而是保留给具体的虚拟机。在下文中，你将学习如何保留 PCI 设备到虚拟机。

代码

表 1：使用 lspci 命令找到你想要排除的 PCI 接口的 PCI ID

```
BTN:~ # lspci

00:00.0 Host bridge: Intel Corporation Mobile 945GM/PM/GMS, 943/940GML
        and 945GT Express Memory Controller Hub (rev 03)
00:01.0 PCI bridge: Intel Corporation Mobile 945GM/PM/GMS, 943/940GML
        and 945GT Express PCI Express Root Port (rev 03)
00:1b.0 Audio device: Intel Corporation 82801G (ICH7 Family)
        High Definition Audio Controller (rev 01)
00:1c.0 PCI bridge: Intel Corporation 82801G (ICH7 Family)
       PCI Express Port 1 (rev 01)
00:1c.1 PCI bridge: Intel Corporation 82801G (ICH7 Family)
       PCI Express Port 2 (rev 01)
00:1c.2 PCI bridge: Intel Corporation 82801G (ICH7 Family)
        PCI Express Port 3 (rev 01)
00:1c.3 PCI bridge: Intel Corporation 82801G (ICH7 Family)
        PCI Express Port 4 (rev 01)
00:1d.0 USB Controller: Intel Corporation 82801G (ICH7 Family)
```

USB UHCI Controller #1 (rev 01)
00:1d.1 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB UHCI Controller #2 (rev 01)
00:1d.2 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB UHCI Controller #3 (rev 01)
00:1d.3 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB UHCI Controller #4 (rev 01)
00:1d.7 USB Controller: Intel Corporation 82801G (ICH7 Family)
USB2 EHCI Controller (rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 Mobile PCI Bridge (rev e1)
00:1f.0 ISA bridge: Intel Corporation 82801GBM (ICH7-M)
LPC Interface Bridge (rev 01)
00:1f.2 IDE interface: Intel Corporation 82801GBM/GHM (ICH7 Family)
SATA IDE Controller (rev 01)
00:1f.3 SMBus: Intel Corporation 82801G (ICH7 Family) SMBus
Controller (rev 01)
01:00.0 VGA compatible controller: nVidia Corporation GeForce
Go 7950 GTX (rev a1)
03:01.0 FireWire (IEEE 1394): Ricoh Co Ltd R5C832 IEEE 1394 Controller
03:01.1 Generic system peripheral [Class 0805]: Ricoh Co
Ltd R5C822 SD/SDIO/MMC/MS/MSPro Host Adapter (rev 19)
03:01.2 System peripheral: Ricoh Co Ltd R5C843 MMC
Host Controller (rev 01)
03:01.3 System peripheral: Ricoh Co Ltd R5C592 Memory
Stick Bus Host Adapter (rev 0a)
03:01.4 System peripheral: Ricoh Co Ltd xD-Picture
Card Controller (rev 05)
09:00.0 Ethernet controller: Broadcom Corporation NetXtreme
BCM5752 Gigabit Ethernet PCI Express (rev 02)
0c:00.0 Network controller: Broadcom Corporation BCM4328
802.11a/b/g/n (rev 01)

(作者：Sander van Vugt 译者：唐琼瑶 来源：TechTarget 中国)

Xen 环境下的内存与 CPU 分配：性能优化

在 Xen 环境下，硬件的管理并不是仅仅告诉它可以使用哪些 PCI 设备就可以完事。在准虚拟环境下，内存与 CPU 分配是可以动态改变的。通过动态更改内存与 CPU 分配，可以达到优化虚拟机性能的目的。在本文中，TechTarget 中国的特约虚拟化专家 Sander van Vugt 将描述这方面的所有信息。

启动物理服务器时，所有的内存资源都会默认被分配给 Domain0。然后，其它虚拟机启动时，会从 Domain0 获取内存资源。如果虚拟机是在完全虚拟化模式下运行，hypervisor 将无法与虚拟内核对话，当前的内存分配也将无法改变。不过，如果是在准虚拟化模式（paravirtualization）下，Xen 的 hypervisor 就可以动态更变内存分配。采用准虚拟化模式时，一定要确保 Domain0 至少可以分得一定的内存资源，以免它内存不足。对于 Domain0 的内存分配最小值，我建议设置为 512MB。

要为 Domain0 预留内存，可以为内核添加一个启动项：dom0_mem= 。例如，dom0_mem=512M。打开 Grub 配置文件进行此设置。在 Grub 配置文件中，你会看到启动 Xen 内核的启动项。它大体如下：

```
title XEN
root (hd0,0)
kernel /xen.gz
module /vmlinuz-
2.6.16 .46-0.14-xen root=/dev/system/root vga=0x314
resume=/dev/system/swap splash=silent showopts
module /initrd-
2.6.16 .46-0.14.xen
```

在此配置文件中的第一个“module”行后面添加 dom0_mem 启动项。添加之后应该是这样：

```
title XEN
 root (hd0,0)
 kernel /xen.gz
 module /vmlinuz-
 2.6.16 .46-0.14-xen root=/dev/system/root vga=0x314
```

```
        resume=/dev/system/swap splash=silent showopts
dom0_mem=512M
      module /initrd-
      2.6.16 .46-0.14.xen
```

设置好 Domain0 的内存分配后，就可以管理你的虚拟机内存分配了。启动一个虚拟机时，通常它会从 Domain0 获取内存资源。内存一旦分配给虚拟机，Domain0 将无法再收回，即使所有虚拟机都被停止也不能收回。正是因为这个原因，所以为 Domain0 设置内存最小值非常重要。

要想更改虚拟机的内存分配，可以利用两个 xm 命令：

- xm mem-set：此命令可以更改一台虚拟机的当前内存分配；
- xm mem-max：此命令可以限定一台虚拟机的内存使用最大值。不过，更改最大值之后需要重启才能生效。

更改内存分配之后，一定要使用 xm list 命令检查设置是否生效和正确：

名字	ID	内存	虚拟CPU	状态	Time(s)
Domain-0	0	898	2	r----	106.1
main		512>	1		16.1
oes2cursus	1	600	1	-b---	21.7
oes2l	2	512	1	-b---	1.0
sled10		256	1		85.1
sles10		512	1		0.0
sles10-1		512	1		0.0
windows2003		256	1		2.7

CPU 管理

与内存一样，你也可以管理虚拟机的 CPU 分配。如果虚拟机使用的是准虚拟化，CPU 的分配也可以动态更改。为虚拟机分配 CPU 时，不一定要根据服务器中的物理 CPU 数目来分。如果你愿意，是可以这么做。不过，这样做是绝对优化不了性能的。如果将虚拟机与指定的物理 CPU 绑定，会帮助你大大地提高虚拟机性能。除此之外，还可以调整 CPU 的运行队列（run queue），使某台虚拟机在 CPU 中具有更高的优先级。

所有可运行的虚拟 CPU（VCPU）都是由物理 CPU 中的本地运行队列管理的。这个队列是按优先级进行排序的，队列中的每个 VCPU 平分 CPU 资源。VCPU 的优先级状态有两种值：over 和 under。Over 表示它占用的 CPU 资源超过了资源平分值，under 表示低于这个平分值。如果 VCPU 的当前状态为 under，调度程序下次则会优先服务该 VCPU。如果调度程序发现在其 CPU 上没有虚拟机为 under 状态，则会看其它 CPU 中是否有 VCPU 状态为 under，如果发现，则立即服务该 VCPU。通过这种方式，所有 CPU 都会平均分配 CPU 资源。

通过设置 weight 和 cap 参数值，管理员可以管理 CPU 的优先级。Weight 参数用于分配 CPU cycle，是一个相对值。一个 weight 为 128 的 VCPU 比一个 weight 为 64 的 VCPU 获得的 CPU cycle 多一倍。因此，利用这个参数可以决定哪个 VCPU 获得更多，哪个获得更少。第二个设置 CPU 的参数是 cap，它设置的是 domain 获得的 CPU cycle 百分数，是一个绝对值。如果设置为 100，就表示那个 VCPU 会 100%地占用物理 CPU 的可用 cycle。如果 cap 为 50，则表示该 VCPU 占用的 CPU cycle 绝不会超过总量的一半。

在如下命令示例中， id 为 3 的虚拟机 weight 为 128，允许使用两个物理 CPU 的所有 CPU cycle：

```
xm sched-credit -d 3 -w 128 -c 200
```

对于虚拟 CPU，还要做的一个重要工作就是 CPU 分配。默认情况下，虚拟 CPU 与物理 CPU 是没有固定联系的。要提高性能，就需要为它们建立一个这样的联系，这个工作很简单易行。为虚拟 CPU 和物理 CPU 建立“联系”的主要好处是可以防止虚拟 CPU 到处游荡。如果没有“联系”，调度程序会为虚拟 CPU 选择一个物理 CPU。当某个物理 CPU 处于繁忙状态时，虚拟 CPU 就会被转移，由另一个物理 CPU 服务。这个工作对性能的影响是很大的。因此，将虚拟 CPU 与物理 CPU 绑定是个不错的办法。

绑定虚拟 CPU 时，首先利用 xm list 命令查看当前配置。然后，在你要查看 CPU 详细信息的 domain 中使用 xm vcpu-list 命令，命令输出结果如下：

```
lin:~ # xm vcpu-list 2
```

Name	ID	VCPU	CPU	State	Time(s)	CPU Affinity
oes21	2	0	1	r--	3693.8	any cpu

这个命令显示，ID 2 domain 当前使用了一个 CPU（ID 0），该 CPU 当前在物理 CPU 0。为了确认它的状态，可以使用如下命令：

xm vcpu-pin 2 0 1

如果你再使用 xm vcpu-list 命令，你会看见 CPU Affinity 由原来的“any cpu”变为了 CPU 1。

注意，这个设置是无法被写入的。意思是，每次重启虚拟机之后，你都必须再重新设置。

最后，你还可以更改虚拟机分配的 CPU 数量。要更改此设置，既可以利用虚拟机管理器（Virtual Machine Manager）进行，也可以使用 xm vcpu-set 命令。例如，将 domain 1 分配的 VCPU 数改为 4 个，则：

xm vcpu-set 1 4

使用该命令时，你会发现它有时不起作用。这是因为，虚拟机的操作系统还必须支持动态更改 CPU 数量，不然就不能这样更改了。所以，在虚拟机的配置文件中更改其 VCPU 数更有效，而且不会因为重启虚拟机而失效。

综述

对于虚拟机的性能优化，内存与 CPU 设置很重要，本文已阐述了其原因。此外，你还了解了如何调整虚拟机在物理 CPU 中的优先级。

（作者：Sander van Vugt 译者：涂凡才 来源：TechTarget 中国）

Xen：完全虚拟化与准虚拟化的硬件驱动管理

虚拟机管理员面对的最困难的任务之一就是硬件管理。硬件管理是一个令人迷惑的话题，因为有时指的是真正的硬件，有时指的是虚拟硬件。有些情况下，使用的是完全虚拟化，有时使用的又是其它虚拟化。因此，根据虚拟化技术的不同，所表示的硬件也不同。在本文中，TechTarget 中国的特约虚拟化专家 Sander van Vugt 将解释完全虚拟化与准虚拟化环境下硬件驱动的不同之处。本文将以 Xen 环境为例。

在 Xen 架构中，驱动域（driver domain）通常为 Domain0。这就意味着只有这个 Domain0 被允许使用“真正的”驱动与硬件设备直接对话。通常，Domain0 被用作驱动域，不过这并不是唯一的一种可能性。在后面你将了解到，如果需要的话，你可以将某个驱动的管理交给某个 Domain。所有不是驱动域的 Domain 都使用虚拟设备驱动，这些虚拟设备驱动会使用驱动域中的驱动。

在 Xen 驱动模块中，有一个后端（back-end）驱动运行于驱动域（通常为 Domain0），然后驱动域直接与硬件设备对话。在 DomainU 中，则有一个前端（front-end）驱动与驱动域中的后端驱动对话。为了与后端驱动通信，Xen 建立了 Xen 总线（bus）。这是一个虚拟总线，所有前端驱动都可以通过它与后端驱动通信。后端驱动的任务就是处理与硬件设备之间的通信。

前端驱动与后端驱动的通信有两种方式：准虚拟化（paravirtualization）和模拟(emulation)。首先，可以采用准虚拟化，当准虚拟化不可用时，可以采用模拟。

如果使用的是准虚拟化的操作系统，也就会自动使用准虚拟化驱动。在这种情况下，驱动软件能识别虚拟化环境。因此，驱动可以发出最优的指令与硬件直接对话。

准虚拟化驱动并不只是在准虚拟化环境才可用，在完全虚拟化中也可以使用。比如说，Novell Virtual Machine Driver Pack 对有些操作系统可用，所以有可能使用该驱动包中的准虚拟化驱动去驱动块设备和网络设备。由于这些设备都会导致超高负载，使用这些驱动会大大地提高性能。

完全虚拟化环境中的驱动可以替代准虚拟化驱动。因为，在这些环境中，驱动自身无法意识到自己是虚拟的，它发出的所有指令都必须被截留和模拟。要完成这个工作，可以使用 Qemu 解决方案。Qemu 是一个处理器模拟器，提供了多种解决方案与设备通信，如模拟磁盘设备。

为了协助完全虚拟化中的驱动，通信必须经过驱动域中的驱动，而这会给性能带来负面的影响。不幸的是，驱动域并没有多余的内存资源来处理这些模拟驱动引起的负载，这是由于驱动域中使用了一种叫做“影子页表（shadow pages）”的特殊技术。

驱动属性

在Xen虚拟化环境下，驱动的属性取决于你所使用的虚拟化技术。因此，有些操作可能在完全虚拟化中可以进行，而在准虚拟化环境下不能。有些问题可能在某种环境中存在，而在另一种环境中没有。在下面的分段中，你将了解到在特定虚拟化环境下设备类型及其属性的相关重要信息。

块设备：通过准虚拟化处理块设备时，你会在DomainU中以xvd（Xen虚拟磁盘）设备的形式看见它们。第一个准虚拟化块设备是xvda，第二个是xvdb，依此类推。

通常，虚拟机中的操作系统会把这些设备看作 SCSI 磁盘设备。在虚拟机中，你最多可以有 16 个这样的设备。虚拟机中的这些 xvd 设备会与 Domain0 中的 xenblk 模块进行对话，虚拟机本身使用了 blkbk 与 blktap 内核模块。

如果你使用的完全虚拟化 DomainU 没有使用准虚拟化驱动，那么在这些虚拟机中会看到“正常的”块设备。也就是说，如果是虚拟的 Linux，你会看见 hda 和 hdb，与非虚拟机上一样。你应该意识到，你的选择要么是这个要么是那个。如果你的某些块设备是完全虚拟的，那么所有的块设备就都必须采用完全虚拟化。

LAN设备：完全虚拟与准虚拟LAN设备的不同之处不如块设备那么明显。两种情况下你都最多只能有三个虚拟网络插件板（network board）。在准虚拟化环境下，netbk和netloop内核模块用于与Domain0 中的xennet模块对话。网卡本身作为xennet网卡使用。使用完全虚拟化时，你可以在Realtek 8139、AMD PCnet32 和NE2000 网络插件板之间进行选择。

视频设备：至于视频板（video board），准虚拟与完全虚拟也是有区别的。一般来说，你是不会在意这个区别的，因为视频板对服务器性能没什么重要影响。使用准虚拟视频板时，它是作为帧缓冲设备使用的。在DomainU中，帧缓冲设备用于提供服务，在前端，该驱动与xenfb模块通信。模拟视频驱动在完全虚拟化环境中被看作Cirrus Logic或一般VESA显卡。

准虚拟化设备驱动的好处

在本文中，我们了解到了 Xen 环境下虚拟驱动使用的两种方法。即便是在完全虚拟化环境，某些设备的准虚拟化驱动使用也是有可能的。这通常会大大地提高设备性能，因为准虚拟设备驱动能识别自己被使用的虚拟化环境。

（作者：Sander van Vugt 译者：涂凡才 来源：TechTarget 中国）

使用 Linux 命令在 Xen 里创建快照

虚拟机快照是一个非常好的功能，它能保存当前虚拟机的状态。不幸的是开源 Xen 不提供对快照的支持，而 Linux 能支持。由于开源 Xen 通常使用 Linux 作为它的特权域，所以你能使用 Linux 命令创建快照。

逐个字节进行快照

在 Xen 里创建快照的一种方法是在保存虚拟机当前状态后使用 Linux dd。这包括以下步骤：

1.使用 xm save 命令禁用当前的虚拟机状态并将其写入磁盘文件。这只把机器状态写入一个文件，不是用于 Xen 磁盘文件或分区的当前状态。使用名称 linux01 这样做，用 xm save linux01 linux01.sav。注意这个命令将停止虚拟机。

2.现在使用 dd 将磁盘镜像文件的当前状态转存到一个备份文件。下面的例子将为 LVM 逻辑卷：

```
dd if=/dev/xenvols/linux01_root      of=/data/xen_linux01_root.img
```

3.使用 xm restore 命令重新启动虚拟机。

这种解决方案的主要缺点在于耗时。dd 命令逐个字节地对虚拟机磁盘文件进行复制，因此需要消费大量时间。所以这种方案不是非常实用。

LVM方法

在 Linux 里，Logical Volume Manager（LVM）也能用于创建快照，它比先前的磁盘文件方法节省了许多时间。这种方法意味着你的虚拟机使用 LVM 逻辑卷作为存储后端，与使用虚拟磁盘文件形成对比。由于这个逻辑卷，你接下来需要创建快照。这个快照是一种备份，只包含当时进行快照时所改变的元数据和块。当你通过元数据使用 dd 制作快照副本时，你通常在原始卷上制作原始块的快照，不需要重新激活原始卷。通过这种方式，能大幅度减少创建快照的时间。步骤如下：

1.使用 xm save 命令保存虚拟机当前状态，并写入磁盘文件：

```
xm save linux01 linux01.sav
```

2.假定你已经有一个 LVM 逻辑卷用来作为你虚拟机的存储后端，使用下面的命令对这个卷进行快照。比较好的准则是使用在原始逻辑卷里所分配磁盘空间的 10%作为快照卷的大小：

```
lvcreate -s -L 1G -n linux01-snap /dev/xenvols/linux01
```

3.由于现在你已经在 LVM 快照里保存了虚拟机的状态，你能重新启动虚拟机，显著减少虚拟机的停机时间：

```
xm restore linux01-sav
```

4.使用 dd 创建虚拟机快照并写入一个镜像文件。由于要使用快照复制所有虚拟机分配的磁盘块，这将花费很长时间：

```
dd if=/dev/xenvols/linux01-snap of=/data/xen01.img
```

5.不要忘记在最后一步移除快照。这很重要，因为快照最终将被完全覆盖而导致快照不能用。这样的问题是将阻止你从原始卷重新启动，因此不要忘记这最后一步：

```
lvremove /dev/xenvols/linux01-snap
```

目前，没有任何一种 Linux 版本提供了在开源 Xen 堆栈里创建虚拟机快照的方法，在本文中，我们学习了使用标准 Linux 工具，如 LVM 和 dd 命令来创建快照。

（作者：Sander van Vugt 译者：唐琼瑶 来源：TechTarget 中国）