HEISON CHAK

virtualizing Asterisk

Heison Chak is a system and network administrator at SOMA Networks. He focuses on network management and performance analysis of data and voice networks. Heison has been an active member of the Asterisk community since 2003.

heison@chak.ca

AS CPU AND MEMORY MODULES BEcome faster and more affordable, setting up what we used to call a "big box" to host multiple virtual machines (VMs) for server consolidation and still achieve close-to-native performance is becoming a reality without costing a fortune.

Although commercial virtualization products such as VMWare, Parallels, and Solaris Zones focus on delivering ease of deployment and administration of VMs, they all have their limitations, ranging from minimum CPU requirements to supported OSes. Xen, however, is an open source virtualization software that is designed with goals similar to those of some commercial products. It also has certain limitations, the biggest one being its inability to run a nonmodified OS (with the kernel being the issue), which has made running the Windows OS impossible in the past. This shortcoming has been addressed with the added support of Intel VT-enabled CPUs or the AMD Pacifica equivalent. (See the Hand article about hardware virtualization, this issue, p. 21.)

Virtualization

Xen was chosen mainly because of the available support for running Windows as a guest OS, with hardware virtualization. The object was to find a virtualization platform that enabled Windows, Linux, and Solaris operating systems. My initial reasons for virtualization were:

- To reduce energy consumption
- The ability to bring up a test environment within minutes
- To stage and test Asterisk 1.4

Until recently, SCSI drives have always been chosen for their performance and reliability. With Serial ATA (SATA) drives becoming more prevalent and larger in capacity, the gap is closing. Replacing an array of RAID 5 SCSI drives with big mirrored SATA drives can increase capacity and reduce electricity consumption. The reduced spindles of SATA drives will generate less heat, reducing cooling requirements.

Building the Virtualization Host

To get started, I installed Xen from source onto a Linux box running Debian Sarge. I then patched a Linux 2.6 kernel with Xen modifications and compiled as a dom0 kernel (for the host OS). After booting the Linux box with the newly built dom0 kernel, you can build a domU kernel (for the guest OS) and a virtual machine template based on Debian. This template can be used to quickly deploy new Linux VMs.

Whereas the kernel for domU remains under /boot of dom0, the root (/) of the VMs resides in image files (.img). Each image file has an ext3 file system and contains a Debian install:

```
vm:/# ls -l /boot/*xen[0U]
-rw-r—r— 1 root root 2347714 2006-11-27 22:57 /boot/vmlinuz-2.6.16.29-
   xen0
-rw-r—r— 1 root root 1263925 2006-11-27 14:40 /boot/vmlinuz-2.6.16.29-
   xenU

vm:/# ls -l /vserver/images/
-rw-r—r— 1 root root 4194304000 2006-10-30 21:54 debian01.img
-rw-r—r— 1 root root 2097152000 2006-10-30 21:56 debian01-swap.img
-rw-r—r— 1 root root 4194304000 2006-10-30 23:35 debian02.img
-rw-r—r— 1 root root 2097152000 2006-10-30 23:38 debian02-swap.img
-rw-r—r— 1 root root 4194304000 2006-10-28 08:43 debian_base.img
-rw-r—r— 1 root root 2097152000 2006-10-28 09:50 debian_base-
   swap.img
```

Deploying a new VM involves duplicating debian_base.img and debian_base-swap.img image files.

See http://www.howtoforge.com/debian_sarge_xen_3.0.3 for step-by-step instructions on how to install Xen on a Debian Sarge system.

Asterisk and Xen

Two VMs have been set up to test Asterisk 1.2 and 1.4, with 128 MB of memory and one virtual CPU assigned to each VM:

```
vm:/# xm list
Name            ID    Mem(MiB)   VCPUs      State     Time(s)
Domain-0        0     374        1 r——      200.4
asterisk-12     1     128        1 -b——     20.6
asterisk-14     2     128        1 -b——     13.1
```

Both virtual machines have access to the Internet to check out the latest source of Asterisk via subversion. Compiling libpri and asterisk was relatively easy; as one might guess, the tricky part is with the Zaptel drivers. These are loadable kernel modules that may need access to hardware, specialized PCI interfaces that communicate with the PSTN. Aside from CPU and memory, virtualizing hardware is more involved and sometimes difficult or impossible.

With Xen 3.0.3, PCI devices can be assigned solely to a VM (domU) but it is not used (or hidden) in the host OS (dom0). ("Tiger" refers to the Zaptel card.)

```
vm:/# lspci | grep -i tiger
00:09.0 Network controller: Tiger Jet Network Inc. Tiger3XX Modem/ISDN
   interface
00:0a.0 Communication controller: Tiger Jet Network Inc. Tiger3XX
   Modem/ISDN interface
00:0c.0 Communication controller: Tiger Jet Network Inc. Tiger3XX
   Modem/ISDN interface

vm:/# cat /etc/xen/debian01-config.sxp
name="asterisk-12"
```

```
kernel="/boot/vmlinuz-2.6-xenU"
root="/dev/hda1"
memory=128
disk=['file:/vserver/images/debian01.img,hda1,w',
      'file:/vserver/images/debian01-swap.img,hda2,w']

# Assign FXO interface to asterisk-12 domU
pci = [ '00:0a.0' ]

# network
vif=[ '' ]
dhcp="off"
ip="10.155.200.1"
netmask="255.255.0.0"
gateway="10.155.1.1"
hostname="asterisk-12.ykz.zealnetworks.com"

extra="3"
```

By assigning a PCI to a domU, one can build Zaptel drivers and use the FXO interface to make calls to the PSTN or to use it as a timing device for Meetme conference.

Limitation with ztdummy

Ideally, a virtual machine should not remain hardware-independent. In Asterisk, conferencing requires a reliable clock to mix audio. Such a timing source is usually featured in the Digium hardware PCI cards, or one can choose to use the ztdummy driver. In Linux 2.6, ztdummy defaults to using the kernel's real-time clock, which is not available to a Xen domU. Falling back to using the USB clock (i.e., OHCI_UCD) didn't work either, as USB support in domU seemed lacking at the time this article was written.

Until ztdummy works properly in domU, one may need to assign the PCI interface to the virtual machine if conferencing is required. Hardware dependency for a virtual machine may not be elegant, but it works.