GNU/Linux Administration - Feature #371

Setting Up Xen on Debian

04/19/2014 09:29 AM - Daniel Curtis

Status:	Suspended	**Start date:**	04/19/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:		**Estimated time:**	2.00 hours
Target version:		**Spent time:**	0.00 hour

Description

Xen is an open-source (GPL) type-1 or baremetal hypervisor, which makes it possible to run many instances of an operating system or indeed different operating systems in parallel on a single machine (or host)

Some of Xen's key features are:

- **Small footprint and interface** (is around 1MB in size). Because Xen uses a microkernel design, with a small memory footprint and limited interface to the guest, it is more robust and secure than other hypervisors.
- **Operating system agnostic**: Most installations run with Linux as the main control stack (aka "domain 0"). But a number of other operating systems can be used instead, including NetBSD and ?OpenSolaris.
- **Driver Isolation**: Xen has the capability to allow the main device driver for a system to run inside of a virtual machine. If the driver crashes, or is compromised, the VM containing the driver can be rebooted and the driver restarted without affecting the rest of the system.
- **Paravirtualization**: Fully paravirtualized guests have been optimized to run as a virtual machine. This allows the guests to run much faster than with hardware extensions (HVM). Additionally, Xen can run on hardware that doesn't support virtualization extensions.

See the Xen Overview on the Xen wiki for more information.

Guest types

Xen supports running two different types of guests: Paravirtualization (PV) and Full or Hardware assisted Virtualization (HVM). Both guest types can be used at the same time on a single Xen system. It is also possible to use techniques used for Paravirtualization in an HVM guest: essentially creating a continuum between PV and HVM. This approach is called PV on HVM. Again see the Xen Overview on the Xen wiki for more information.

Domain 0

Xen has a special domain called domain 0 which contains drivers for the hardware, as well as the toolstack to control VMs. Domain 0 is often referred to as dom0.

Domain 0 (Host) Installation

Initial Host Installation

Before installing Xen you should install Debian on the host machine. This installation will form the basis of Domain 0.

Installing Debian can be done in the usual way using the DebianInstaller. See the Debian Release Notes for more information on installing Debian.

In order to install Xen you will either a 32-bit PC (i386) or 64-bit PC (amd64) installation of Debian. Although it is recommended to always run a 64-bit hypervisor note that this does not mean one has to run a 64-bit domain 0. It is quite common to run a 32-bit domain 0 on a 64-bit hypervisor (a so-called "32on64" configuration).

In general you can install your Domain 0 Debian as you would any other Debian install. The main thing to consider is the partition layout since this will have an impact on the disk configurations available to the guests. The Xen wiki has some Host OS Installation Considerations which may be of interest. To paraphrase that source: if your Domain 0 Debian system will be primarily used to run guests, a good rule of thumb is to set aside 4GB for the domain 0 root filesystem (/) and some swap space (swap=RAM if RAM<=2GB; swap=2GB if RAM>2GB). The swap space should be determined by the amount of RAM provided to Dom0, see Configure Domain 0 Memory

Use the rest of the disk space for a LVM physical volume.

If you have one disk, the following is a reasonable setup: create 3 physical partitions: sda1, sda2, sda3. The root (ext4) and swap will be on the first two and the remainder will be under Logical Volume Management (lvm). With the LVM setup, create 1 physical volume and then one volume group. Give the volume group a name, such as `vg0'.

Installing Xen Packages

The Xen and debootstrap software in Squeeze (Debian 6.0) are very much newer than that in Lenny. Because of that, working with Xen becomes a lot easier.

The setup described here is tested for Debian Squeeze and Ubuntu Maverick virtual machines, but should work for a lot more.

First install the hypervisor, xen aware kernel and xen tools. This can be done by a metapackage:

```
apt-get install xen-linux-system
```

Checking if our host support HVM

If you want to use Hardware assisted Virtualization you must verify if your CPU supports this technology. To do this run this commmand:

```
egrep '(vmx|svm)' /proc/cpuinfo
```

If supported, egrep will return VMX for Intel and SVM for AMD.

To get Xen HVM support on Squeeze the qemu device model package, which provides the necessary emulation infrastructure for an HVM guest, is also required:

```
apt-get install xen-qemu-dm-4.0
```

This is no longer needed in Wheezy since the device model is part of the Xen packages.

Prioritize Booting Xen Over Native

Debian Squeeze uses Grub 2 whose default is to list normal kernels first, and only then list the Xen hypervisor and its kernels.

You can change this to cause Grub to prefer to boot Xen by changing the priority of Grub's Xen configuration script (20_linux_xen) to be higher than the standard Linux config (10_linux). This is most easily done using dpkg-divert:

```
dpkg-divert --divert /etc/grub.d/08_linux_xen --rename /etc/grub.d/20_linux_xen
```

- To undo this:

  ```
  dpkg-divert --rename --remove /etc/grub.d/20_linux_xen
  ```

After any update to the Grub configuration you must apply the configuration by running:

```
update-grub
```

Configure Networking

In order to give network access to guest domains it is necessary to configure the domain 0 network appropriately. The most common configuration is to use a software bridge.

It is recommended that you manage your own network bridge using the Debian network bridge. The Xen wiki page Host Configuration/Networking also has some useful information. The Xen supplied network scripts are not always reliable and will be removed from a later version. They are disabled by default in Debian's packages.

If you have a router that assigns ip addresses through dhcp, the following is a working example of the /etc/network/interfaces file using bridge-utils software.

```
#The loopback network interface
auto lo
iface lo inet loopback

iface eth0 inet manual

auto xenbr0
iface xenbr0 inet dhcp
   bridge_ports eth0

#other possibly useful options in a virtualized environment
  #bridge_stp off       # disable Spanning Tree Protocol
  #bridge_waitport 0    # no delay before a port becomes available
  #bridge_fd 0          # no forwarding delay
```

Other configuration tweaks

Configure Domain 0 Memory

By default on a Xen system the majority of the hosts memory is assigned to dom0 on boot and dom0's size is dynamically modified ("ballooned") automatically in order to accomodate new guests which are started.

However on a system which is dedicated to running Xen guests it is better to instead give dom0 some static amount of RAM and to disable ballooning.

The following examples use 1024M.

In order to do this you must first add the dom0_mem option to your hypervisor command line. This is done by editing /etc/default/grub and adding

```
# Xen boot parameters for all Xen boots
GRUB_CMDLINE_XEN="dom0_mem=1024M"
```

at the bottom of the file.

Note: On servers with huge memory, Xen kernel crash. You must set a dom0 memory limit. Take care on Wheezy, 1024M is not enough and cause kernel crash at boot with out-of-memory message.

Remember to apply the change to the grub configuration by running update-grub!

Then edit /etc/xen/xend-config.sxp to configure the toolstack to match by changing the following settings:

```
(dom0-min-mem 1024)
(enable-dom0-ballooning no)
```

At this point you should reboot so that these changes take effect.

Configure dom0 CPUs

There are some useful tweaks of dom0 cpu utilization.

By default all CPUs are shared among dom0 and all domU (guests). It may broke dom0 responsibility if guests consume too much CPU time. To avoid this, it is possible to grant one (or more) processor core to dom0 and also pin it to dom0.

Add following options to /etc/default/grub to allocate one cpu core to dom0:

```
dom0_max_vcpus=1 dom0_vcpus_pin
```

Make such changes in /etc/xen/xend-config.sxp:

```
(dom0-cpus 1)
```

Configure guest behaviour on host reboot

By default, when Xen dom0 shuts down or reboots, it tries to save (i.e. hibernate) the state of the domUs. Sometimes there are problems with that - it could fail because of a lack of disk space in /var, or because of random software bugs. Because it is also clean to just have the VMs shutdown upon host shutdown, if you want you can make sure they get shut down normally by setting these parameters in /etc/default/xendomains:

```
XENDOMAINS_RESTORE=false
XENDOMAINS_SAVE=""
```

Configure Boot Parameters

You may also want to pass some boot parameters to Xen when starting up in normal or recovery mode. Add these variables to /etc/default/grub to achieve this:

```
# Xen boot parameters for all Xen boots
GRUB_CMDLINE_XEN="something"
# Xen boot parameters for non-recovery Xen boots (in addition to GRUB_CMDLINE_XEN)
GRUB_CMDLINE_XEN_DEFAULT="something else"
```

Remember to apply the change to the grub configuration by running update-grub!

More information on the available hypervisor command line options can be found in the upstream documentation.

Configure PCI pass-through Parameters

This information is incomplete for Squeeze and needs to be updated for Wheezy

To enable PCI pass-through, you need to know the BDF (Bus, Device, Function) id of the device. This is obtained through the lspci command, with the output containing the BDF in the format: (BB:DD.F) at the start of the line. To hide a device from Dom0 you will need to pass these boot parameters to Xen when starting. For example if using a Dom0 with 512M of memory and two devices at 01:08.1 and 01:09.2, add these variables to /etc/default/grub to achieve this:

```
# Xen boot parameters for all Xen boots
GRUB_CMDLINE_XEN="dom0_mem=512M pciback.hide=(01:08.1)(01:09.2)"
# Xen boot parameters for non-recovery Xen boots (in addition to GRUB_CMDLINE_XEN)
GRUB_CMDLINE_XEN_DEFAULT="something else"
```

for Squeeze use "pciback.hide" (kernels < 2.6.32.10), for Wheezy (I have not tested this yet) use "xen-pciback.hide"

for Squeeze you need to pass all of the devices on the bus, eg to pass any device on the 01:DD.F bus, you have pass all of them: (01:08.1)(01:09.2)(01:09.3)etc.

Remember to apply the change to the grub configuration by running update-grub!

At least in Wheezy (not tested in Squeeze) the xen-pciback module needs to be configured through modprobe.conf and added to the initramfs additionally.

Configure the xen-pciback module by adding a modprobe include file (e.g. /etc/modprobe.d/xen-pciback.conf) with the following content (given that the PCI device would be assigned to module e1000e normally):

```
install e1000e /sbin/modprobe xen-pciback; /sbin/modprobe --first-time --ignore-install e1000e
options xen-pciback hide=(0000:03:00.0)
```

Add the xen-pciback module to initramfs by adding it to /etc/initramfs/modules and running update-initramfs -u afterwards.

Please note that pci-passthrough is broken when msi is enabled (default) in Linux kernels < 3.14. Use Linux kernel >= 3.14 in DomU/VM or set pci=nomsi for DomU/VM kernel as workaround. See the following thread for detailed information: http://thread.gmane.org/gmane.comp.emulators.xen.user/81944/focus=191437

Enable Serial Console

To get output from GRUB, the Xen hypervisor, the kernel and getty (login prompt) via both VGA and serial console to work, here's an example of the right settings on squeeze:

Edit /etc/default/grub and add:

```
GRUB_SERIAL_COMMAND="serial --unit=0 --speed=9600 --word=8 --parity=no --stop=1"
GRUB_TERMINAL="console serial"
GRUB_TIMEOUT=5
GRUB_CMDLINE_XEN="com1=9600,8n1 console=com1,vga"
GRUB_CMDLINE_LINUX="console=tty0 console=hvc0"
```

Here's what I used to configure the serial console (for a Supermicro X8STi-F motherboard with IPMI and SOL):

```
GRUB_CMDLINE_XEN="loglvl=all guest_loglvl=all com1=115200,8n1,0x3e8,5 console=com1,vga"
GRUB_CMDLINE_LINUX="console=hvc0 earlyprintk=xen"
```

In /etc/inittab you need at least these lines:

```
1:2345:respawn:/sbin/getty 38400 hvc0
2:23:respawn:/sbin/getty 38400 tty1
# NO getty on ttyS0!
```

This way, tty1 will show up at the VGA output, and the hvc0 will show up at the serial console.

To keep both Xen and dom0 kernel output on the same tty, just omit the "vga"-related settings from the above setup.

If you need to debug Xen and see a crash dump of the kernel, you can do it using IPMITool if your server has SOL:

```
ipmitool -I lanplus -H server-ip-address -U your-username sol activate | tee my-log-file.txt
```

Installation as a DomU (guest)

Using xen-tools

xen-tools is a set of scripts which can easily create fully configured Xen guest domains.

Once you have installed dom0 you can install xen-tools on your host with:

```
apt-get install xen-tools
```

To configure xen-tools, you can edit /etc/xen-tools/xen-tools.conf which contains default values that the xen-create-image script will use. The xen-create-image(8) manual page contains information on the available options.

To give a different path where the domU images being saved and enable the superuser password in the initial build, we will edit the /etc/xen-tools/xen-tools.conf file and uncomment this lines:

```
dir = /home/xen/
passwd = 1
```

Then you can create virtual machines with this command:

```
xen-create-image --hostname <hostname> --ip <ip> --vcpus 2 --pygrub --dist <lenny|squeeze|maverick
|whatever>
```

To start the created VM run the command:

```
xm create /etc/xen/virtual_machine.cfg
```

To erase a VM image (even the main directory) run the command:

```
xen-delete-image VMs_name
```

Possible problems and bugs

- If your domU kernel happens to miss support for the xvda* disk devices (the xen-blkfront driver), use the --scsi option that makes the VM use normal SCSI HD names like sda*. You can also set scsi=1 in /etc/xen-tools/xen-tools.conf to make this the default.
- When using xen-tools with --role on Squeeze be aware of #588783: 'Should mount /dev/pts when creating image'. This is fixed in Wheezy.

Using Debian Installer

The Xen wiki page Debian Guest Instalation Using DebianInstaller contains instructions on how to install Xen DomU from Lenny onwards using ?Debian Installer.

Upgrading/transition

See also: Debian Release Notes

Upgrading a server to Squeeze that uses both Lenny Dom0 and DomU's is fairly straightforward. There are a few catches that one needs to be aware of however: Reference

1. Dom0 Issues
   - The Xen packages will not upgrade themselves. They must be manually removed and the latest Xen packages must be installed from the Debian Squeeze repository through apt.
   - xen.independent_wallclock sysctl setting is not available for newer squeeze kernels supporting pvops. If you have relied on it, you would have to run ntpd in dom0 and domUs. source
2. DomU Issues
   - A Squeeze DomU will not be able to boot on the Xen-3.2 package supplied by Lenny because this older version will not support grub2. A Lenny DomU can be upgraded to Squeeze while running on a Lenny Dom0 but it will not be able to be booted until the Dom0 has been upgraded to the Xen-4.0 packages.
   - The entries added to chain load grub1 to grub2 will not allow pygrub to find the correct partition. Before rebooting a freshly upgraded Squeeze DomU, make sure to rename or remove /boot/grub/menu.lst. This will force pygrub to look for the /boot/grub/grub.cfg file which will be in the correct format.
   - A qcow image created with qcow-create and the BACKING_FILENAME option on Lenny will not be able to boot on Squeeze because the ability to use qcow images as backing files has been removed in Xen versions after 3.2. Also, if you try to boot such an image on Squeeze, Xen will silently convert the qcow images L1 table to big endian (you'll find "Converting image to big endian L1 table" in the logfiles), effectively rendering the image unusable on both Squeeze and Lenny!

Note on kernel version compatibility

The new 2.6.32 kernel images have paravirt_ops-based Xen dom0 and domU support.

When you create an image for a modern Debian or Ubuntu domU machine, it will include a kernel that has pv_ops domU support, it will therefore not use a Xen kernel, but the "stock" one, as it is capable of running on Xen's hypervisor.

Common Errors

dom0 automatic reboots

{i} Note: if Xen is crashing and reboot automatically, you may want to use noreboot xen option, to prevent it from rebooting automatically.

Edit /etc/default/grub and add the "noreboot" option to GRUB_CMDLINE_XEN, for example:

```
GRUB_CMDLINE_XEN="noreboot"
```

Error "Device ... (vif) could not be connected"

You need to configure some basic networking between dom0 and domU.

The recommended way to do this is to configure bridging in /etc/networking/interfaces. See BridgeNetworkConnections and/or the Xen wiki page Host Configuration/Networking for details.

{i} Note: The use of the 'network-script' option in /etc/xen/xend-config.sxp is no longer recommended.

'clocksource/0: Time went backwards'

If a domU crashes or freezes while uttering the famous lasts words 'clocksource/0: Time went backwards' see Xen/Clocksource.

PV drivers on HVM guest

It may be possible to build the PV drivers for use on HVM guests. These drivers are called unmodified_drivers and are part of the xen-unstable.hg repository. You can fetch the repository using mercurial thus:

```
hg clone http://xenbits.xen.org/xen-unstable.hg
```

The drivers reside under xen-unstable.hg/unmodified_drivers/linux-2.6. The README in this directory gives compilation instructions.

A somewhat dated, detailed set of instructions for building these drivers can be found here:

- http://wp.colliertech.org/cj/?p=653

Resources

- https://wiki.debian.org/Xen

History

#1 - 02/20/2016 07:52 PM - Daniel Curtis

- Status changed from In Progress to Suspended