Xen 4.3 (Tentative) Documentation Summary

The objective is setting up an IOMMU compatible system with Xen and a spare discrete graphics card for passthrough to a virtual machine environment (likely Windows 7/8).

All documentation is Debian Wheezy specific, and may not apply to other Dom0 platforms.

Requirements:

- IOMMU Compatible Hardware
- Debian Wheezy

Preparation:

- Install Debian Wheezy

I omit the desktop environment from my Wheezy installation as it includes tons of software I don't want. However, I do install gnome3 in order to access my virtual machines via VNC or SDL during initial installation and setup.

Key Hardware

These are the three parts I am using worth mentioning:

- ASRock Z77 Extreme9 (http://www.asrock.com/mb/Intel/Z77%20Extreme9/)

- Intel Core i7 3770 (http://ark.intel.com/products/65719/)
- XFX AMD Radeon HD 6870 (http://www.amazon.com/XFX-Radeon-MINIDP-PCI-E-HD687AZDFC/dp/B004O0OKXK/?qid=1373828783&sr=8-5&keywords=AMD+6870)

List of Packages Installed

Basic Packages:

```
sudo ssh parted ntp screen tmux vim git mercurial
```

Basic Packages (Extra):

```
pz7ip-full exfat-fuse exfat-utils
```

Kernel Builder Packages:

```
build-essential libncurses5-dev kernel-package fakeroot
```

Minimalist Gui Packages:

```
gnome-session gnome-terminal gnome-disk-utility gnome-screenshot gnome-scree
```

Gui Packages (Extra):

```
gparted guake eog gnash vlc gtk-recordmydesktop chromium
```

Xen Minimalist Packages:

```
gcc-multilib bcc flex bison git-core mercurial gawk bridge-utils libvncserve
```

Xen Gui Packages:

```
tgif transfig libsdl-dev gvncviewer
```

Xen Documentation Related Packages:

```
texinfo texlive-latex-base texlive-latex-recommended texlive-fonts-extra texl
```

The minimalist list has been checked against Debian Wheezy for dependencies, selecting the top-most items to reduce the size of the list of components. I tend to omit the documentation related Xen packages.

System Configuration

After installing all the software packages, I proceed to configure the system.

Many of the actions I take are actually omitted from these steps, and the ones listed may be unrelated to the Xen platform itself, but worth noting.

I setup the system to automatically update installed packages by creating an executable file `/etc/cron.weekly/aptitude` with these contents:

```
#!/bin/sh
# Weekly Software Update Processing
aptitude clean
aptitude update
aptitude upgrade -y
aptitude safe-upgrade -y
```

Ideally one should wrap the results of the above process into a log file or at least an email for documentation of system changes in case we run into problems.

Since I am using a SSD for my main drives I add discard to the LVM configursation in Dom0:

```
sed -i 's/issue_discards = 0/issue_discards = 1/' /etc/lvm/lvm.conf
```

Then I create another weekly executable file `/etc/cron.weekly/fstrim` with:

```
#!/bin/sh
for mount in / /boot /home /var/log /tmp; do
    fstrim $mount
done
```

The fstrim file runs this process weekly instead of per-operation, since per-operation puts higher CPU demand on the system. Be sure to modify the loop to account for any partitions you have that are trim capable.

I add my public key to authorized hosts on Dom0, and update the SSH port for obfuscation, as well as setting `PasswordAuthentication` to `no`.

I establish a basic set of iptables in `/etc/firewall.conf`:

```
*filter

# Block Loopback
-A INPUT -i lo -j ACCEPT
-A INPUT ! -i lo -d 127.0.0.1/8 -j REJECT

# Accept established inbound connections
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
# Accept Ping
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT

# Accept SSH
-A INPUT -p tcp -m state --state NEW --dport #### -j ACCEPT

# Prevent Bridge Cross-talk
-A FORWARD -i eth0 -o xenbr1 -j REJECT
-A FORWARD -i eth0 -o eth1 -j REJECT
-A FORWARD -i eth1 -o xenbr0 -j REJECT
-A FORWARD -i eth1 -o eth0 -j REJECT

# Allow VNC Ports
-A OUTPUT -p tcp --dport 5000:5020 -j ACCEPT

# Set Remaining Defaults
-A INPUT -j REJECT
-A FORWARD -j ACCEPT
-A OUTPUT -j ACCEPT

COMMIT

*filter
```

You can easily modify the VNC line to accept only an intranet range of addresses if you wish to secure it without revoking network access.

I then set the firewall to load with the network interface by creating an executable file at `/etc/network/if-up.d/` with:

```
#!/bin/sh
iptables -F
iptables-restore < /etc/firewall.conf
```

List of steps temporarily omitted from this guide:

- Patching Guake & setting it to start at login
- Allowing root login
- Adding a locale
- Installing Fonts
- Installing Sublime Text
- Configure git

Building a Kernel

I configure the kernel package service for building by setting the concurrency level in `/etc/kernel-pkg.conf`:

```
echo "\n# Concurrency Level\nCONCURRENCY_LEVEL=$(nproc)" >> /etc/kernel-pkg.c
```

Note the above line for meant for a script, but nproc will give you your CPU's and can be used if you are uncertain as to the number. Also some documentation claims the concurrency can be set to the number of processors plus one (though I do not know whether this is valid).

The latest kernel version I have had success using is 3.9.9, kernel 3.10 has hidden its Xen configuration options during my last attempt.

It is recommended to navigate to a partition with 7GB of spare space, and create a source folder. Then we can download the package & extract the contents:

```
wget --no-check-certificate https://www.kernel.org/pub/linux/kernel/v3.x/linu
tar -xf linux-3.9.9.tar.xz
cd linux-3.9.9
```

Next copy the systems configuration from `/boot/`, if you have a fresh install of Debian Wheezy it should be:

```
cp /boot/config-3.2.0-4-amd64 .config
```

There are a two options as to how to proceed from here. You can modify the `.config` file manually and then run the configuration tools to automatically correct and configure the system. Alternatively you can run `make menuconfig` and navigate to each option manually (the preferred method).

My objective has been to make the process scripted, so I use the unadvised method of adding flags to `.config`, but I recommend that if you are unfamiliar with building a kernel that you take the time to learn how to navigate `make menuconfig`.

```
CONFIG_VIRT_CPU_ACCOUNTING_GEN=y
CONFIG_NUMA_BALANCING=y
CONFIG_PARAVIRT_TIME_ACCOUNTING=y
CONFIG_MOVABLE_NODE=y
CONFIG_X86_IO_APIC=y
CONFIG_ACPI=y
CONFIG_ACPI_PROCFS=y
CONFIG_XEN_DOM0=y
CONFIG_PCI_XEN=y
CONFIG_XEN_DEV_EVTCHN=y
CONFIG_XENFS=y
CONFIG_XEN_COMPAT_XENFS=y
CONFIG_XEN_SYS_HYPERVISOR=y
CONFIG_XEN_GNTDEV=y
CONFIG_XEN_BACKEND=y
CONFIG_XEN_NETDEV_BACKEND=y
CONFIG_XEN_BLKDEV_BACKEND=y
CONFIG_XEN_PCIDEV_BACKEND=y
CONFIG_XEN_PRIVILEGED_GUEST=y
```

```
CONFIG_XEN_BALLOON=y
CONFIG_XEN_SCRUB_PAGES=y
CONFIG_PCI_STUB=y
CONFIG_XEN_WDT=y
CONFIG_XEN_BALLOON_MEMORY_HOTPLUG=y
CONFIG_PARAVIRT=y
CONFIG_XEN=y
CONFIG_PARAVIRT_GUEST=y
CONFIG_PARAVIRT_SPINLOCKS=y
CONFIG_XEN_BLKDEV_FRONTEND=y
CONFIG_XEN_NETDEV_FRONTEND=y
CONFIG_XEN_PCIDEV_FRONTEND=y
CONFIG_INPUT_XEN_KBDDEV_FRONTEND=y
CONFIG_XEN_FBDEV_FRONTEND=y
CONFIG_XEN_XENBUS_FRONTEND=y
CONFIG_XEN_SAVE_RESTORE=y
CONFIG_XEN_GRANT_DEV_ALLOC=y
CONFIG_XEN_TMEM=y
CONFIG_CLEANCACHE=y
CONFIG_FRONTSWAP=y
CONFIG_XEN_SELFBALLOONING=y
```

Note: The documentation regarding these flags indicates that many are for DomU guests. It is my understanding that Dom0 is effectively a privileged DomU, so I add all of the flags. Many of these may not be required, but as far as I know having them has not harmed my system in any way.

- Xen Kernel Reference (http://wiki.xen.org/wiki/Mainline_Linux_Kernel_Configs)

If you run `make menuconfig` you can search for the above flags using `/CONFIG_XEN` and hitting enter, or any of the above titles (they are case sensative). This will help you find where they are located and change them manually.

My approach is to add them to the end of `.config` and run `yes "" | make oldconfig`, which automatically sorts them out. Note that if errors occur you won't get intuitive

feedback and if you added them by hand you may want to grep `.config` to be sure all the right flags have been set.

Next we want to build our kernel and headers:

```
make-kpkg clean
make-kpkg --initrd --revision=3.9.9.$STATE.custom kernel_image kernel_header
```

On an Intel Core i7 IvyBridge 3770 this process takes just under 20 minutes. It will create two `.deb` packages in the parent directory, which can be installed at your discretion (starting with the kernel image, the headers depend on it).

Installation can be done like this:

```
dpkg -i linux-image-3.9.9_3.9.9.dom0.custom_amd64.deb
dpkg -i linux-headers-3.9.9_3.9.9.dom0.custom_amd64.deb
```

There are many other flags I am currently investigating for performance reasons, such as specifying non-generic CPU, and whether `CONFIG_HZ` has noticable effects on performance with a tickless kernel. My documentation here is far from complete. Also the XEN Guest flags that may or may not be required, clarification or tests as to whether they help are planned.

Installing the kernel image should update grub, and you are set to reboot the system before moving onto Xen.

Building Xen

Ideally you will want to change to your source folder and ensure 2GB of free space.

Let's clone Xen and checkout the version:

```
git clone git://xenbits.xen.org/xen.git
cd xen
git checkout -b stable-4.3 origin/stable-4.3
```

Next we want to configure Xen for Debian. First we have to update the Config.mk so it specifies the layout:

```
sed -i "s/^PYTHON_PREFIX_ARG.*/PYTHON_PREFIX_ARG ?= --install-layout=deb/" C
```

Next we set some temporary variables and the configuration script

```
CURL=$(which curl-config)
XML=$(xml2-config)
./configure --enable-githttp
```

Finally we run the build process & generate a `.deb` package:

```
make -j$(nproc) world
make -j$(nproc) debball
```

This process generally takes less time than building the Kernel, and the `.deb` generated can be found in the `dist/` folder.

We can install it via:

```
dpkg -i dist/*.deb
```

Now we have to run through a series of configuration steps.

The xen installation on debian creates three symlinks in /boot to the same file, also in /boot, which in turn generates four records when you run update-grub. I find this to be messy, so I remove them and the xen-symbols file (the below lines are for a script):

```
for FILE in /boot/xen*
do
    if [ -L $FILE ];then
        rm -f $FILE
    fi
done
rm -f /boot/xen-syms*
```

My understanding is that the xen-symbols are used for debugging, but I don't claim to know how that works. What I do know is it creates a failed entry at the top of my grub configuration and prevents the system from booting without interactively selecting the right entry.

Next I disable the SAVE & RESTORE features, these work great but they consume a significant amount of disk space. If you have a small Dom0 disk size and no partition to keep the contents it is best to disable it:

```
sed -i "s/XENDOMAINS_SAVE=\/var\/lib\/xen\/save/XENDOMAINS_SAVE=/" /etc/defau
sed -i "s/XENDOMAINS_RESTORE=true/XENDOMAINS_RESTORE=false/" /etc/default/xen
```

We need to register some init scripts, but there is a trick to avoid errors with xen-watchdog's invalid boot order.

Debian uses insserv, and will translate `update-rc.d` commands, but will ignore the supplies start and stop numbers. The comments inside xen-watchdog indicate it is for xend which has been deprecated, so I don't know how vital it is to have the service running, but there are certainly errors if we just run the commands without checking insserv.

There are two lines `Required-Start` and `Required-Stop`, and we want to add xendomains to the end of each line, like so:

```
sed -i "s/# Required-Start:     $syslog $remote_fs/# Required-Start:     $sys
sed -i "s/# Required-Stop:     $syslog $remote_fs/# Required-Stop:     $syslo
```

Next we want to register three scripts with insserv:

```
update-rc.d xencommons defaults
update-rc.d xendomains defaults
update-rc.d xen-watchdog defaults
```

I generally access the server as a sudo user, not as root, and it would be convenient to not have to prefix xl commands with sudo and enter a password. My solution is to augment the sudoers file:

```
# Allow sudo group passwordless xl execution
%sudo ALL=(ALL:ALL) ALL, !/usr/sbin/xl, NOPASSWD: /usr/sbin/xl
```

Then add this to my .bashrc:

```
# XL Alias
alias xl='sudo xl'
```

Next we need to modify grub to limit dom0 to 4GB of RAM (or less, at your discretion) and hide PCI devices. If you have not already, run lspci and identify any devices you want to pass. Devices I pass include a wireless network card, secondary onboard 4-port SATA controller, AMD Radeon graphics card & audio function, and four USB controllers.

Memory ballooning takes time, and I have encountered problems where without setting

a limit on Dom0 memory starting my DomU guests fails due to unavailable memory. Subsequent attempts to boot the machines works as the initial attempt triggers the balloon request but no effort is made to wait for the memory to become available..

Modify the below commands to suite your needs:

```
mv /etc/grub.d/20_linux_xen /etc/grub.d/09_linux_xen
sed -r -i "s/(multiboot.*)/\1 dom0_mem=4096M dom0_max_vcpus=2/" /etc/grub.d/(
sed -r -i "s/(module.*ro.*)/\1 xen-pciback.hide=(03:00.0)(03:00.1)" /etc/grul
update-grub
```

I used to limit Dom0 CPU's but lately I have stopped. I don't believe there is any benefit in doing so, the boot time of the main system becomes slower and I have seen no benefit from limiting its cpu access.

The last command rebuilds the grub options, and since we moved linux_xen to 09 it will load before the default system kernel.

Finally, according to the latest documentation because of the change in installation directories we want to run `ldconfig` to fix our "dynamic linker cache".

- Compiling Xen (http://wiki.xen.org/wiki/Compiling_Xen_From_Source)

When you reboot the system you should be able to run `xl info` or `xl dmesg` and get non-erroneous output. If you do then Xen has been successfullu installed.

Creating a Windows DomU /w Passthrough

Unfortunately, this process appears to vary greatly from person to person. I cannot gaurantee that these steps will work for everyone, but I will share them since they worked for me.

I start with a fresh reboot of Dom0. The biggest bug I have identified on my system is that if the graphics card has been initialized by any system prior to be passed through to a virtual machine, it changes state and the performance degradation cycle begins.

So far I have not found a resolution to this problem. I have tried fully shutting down a virtual machine, disconnecting pciback and even echoing into the new sysfs reset file, but this does not appear to resolve the problem of an initialized state,

With Xen 4.2 and qemu traditional I used to eject the card to reset it at boot time, and the card would automatically reinstall itself. With Xen 4.3 ejecting the card causes it to disappear from the DomU entirely, and it does not automatically reinitialize. However, if I reboot Windows from VNC or SDL after ejecting the card, it fixes the degradation problem. This work-around requires me to switch monitor output, so it's not exactly ideal, but so far it is the best I have.

My installation process is two steps:

1. Install windows over SDL or VNC with no devices passed.
2. Run through windows updates and install important non-3D dependent software
3. Shutdown Windows and create a backup (dd to file or a second lv)
4. Reboot the physical machine making sure pciback loads for the graphics card and other devices
5. Pass the devices to Windows, preferably manually via pci-attach*
6. Install the drivers, and when asked to reboot shutdown instead
7. Reboot the physical system
8. Add pci devices to Windows configuration file and boot Windows

If things go smoothly your system will have graphics output during Step 6. Ideally you should download something like Passmark benchmark to test whether you experience performance degradation. The performance change will not be noticeable in regular desktop use, only when executing 3D applications.

-
    - Adding the devices in the configuration for windows 7 and especially Windows 8 will automatically install drivers at boot time. I have tried to disable this feature in Windows 8 to no avail, Windows 7 is possible but really a problem. The best solution is to use pci-attach after the system has booted.

Remaining Problems & Questions:

The list below is a summary of the problems and questions I still face with my system. By following my guide, you may face the same list.

- If Dom0 is a privileged DomU, should we be enabling the DomU kernel flags in addition to the Dom0 kernel flags?
- What purposes does xen-watchdog server, is it only for xend, and do we still need it for xl?
- What benefits can be had by limiting Dom0 vcpu's in grub?
- Upstream qemu fails to load virtual machines with VGA passthrough and a large amount of memory (3600MB+ in my case breaks the DomU).
- Does anyone know exactly what Windows device ejection does to the hardware, or how we can do the same from Linux (such as Dom0)?
- Is primary passthrough scheduled to be implemented in Xen 4.4 upstream qemu?