Xen Performance Tunning

"The book of XEN:
A Practical Guide For System Administrator"
C. Takemura, Luke S. Crawford

Chapter 12, Tips, …

Outline

- Introduction
- Beyond Para-virtualization: HVM
- Para-virtualized drivers in HVM (PV-on-HVM)
- Native devices in virtual machines: PCI pass-through
- PVH

Introduction

- Performance overhead reduced by minimizing emulation
  - Para-virtualization
  - Use the existing hardware features
  - The ultimate objective is to keep overhead (extra instructions over a native execution) as low as possible (<5%)

- Para-virtualization limitations
  - Constraints “guest” OS: requires to patch “kernel” guest
    - How to run “Windows” in a domU
  - Don’t take effective use of some interesting HW features
    - EPT, I/O-pass-through

- Alternative methods
  - Full virtualization or HVM

HVM approach

- Boot an unmodified OS
  - Use BIOS emulation (QEMU) to have a fake keyboard, VGA, disk, networt, … (I/O approach #2)
- Advantages of HVM
  - Increase flexibility by exploiting processor acceleration features
  - Support for unmodified OSes
- Disadvantages
  - Guest OS code will raise VMM intervention?
    - Avoided by vmx mode
  - Shadow page table handling?
    - Done in hardware (if supported)
  - Native I/O
    - Potentially done in hardware

How to create a HVM domU

- Config Example

```
#!/bin/python
# HVM EXPECIFICS
# NO APPEND (ROOT)
kernel ="/usr/lib/xen-4.1/boot/hvmloader"
builder = 'hvm'
boot = 'cda' #Boot order # boot on floppy (a), hard disk (c), Network (n) or CD-ROM (d)

#BIOS EMULATION
device_model="/usr/lib/xen-4.1/bin/qemu-dm"
acpi=1 #ENABLE SUPPORT FOR POWER MANAGEMENT
apic=1

#ENABLE VNC FOR ACCESS CONFIGURATION
sdl=0
vnc=1
stdvga=0
serial='pty'

#USUAL BUSSINES
vcpus       ='2'
maxvcpu     =2
memory      ='128'

#
#  Disk device(s).
#
#
disk        = [
                  'file:/xen/domains/prueba/disk.img,hda,w',
                  'file:/xen/netinst.iso,hdb:cdrom,w',
              ]

name        = 'my_hvm'

#
#  Networking (we have to declare the
#  bridge, by default eth0)
#  XEN IS QUITE PICKY HERE!!!
vif = ['bridge=eth0']

#
#  Behaviour
#
on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

How to access a HVM domU?

- Using a graphical front-end in dom0 (default mode)
  - We can proceed as usual, including installation

```
xm create hvm.cfg
Startx
vncviewer 127.0.0.1:0.<number_hvm_machine> if #domU>1
```

  - Sometimes, really clunky

- Using a xen console (recommended mode with no network)
  - Tell QEMU to pass serial output to Xen console
    - `serial='pty'` in cfg
  - Tell GRUB2/kernel (in domU) to pass its messages to serial console. In /etc/defaults/grub (update with update-grub)
    - `GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,38400n8"`
    - `GRUB_TERMINAL=serial`
    - `GRUB_SERIAL_COMMAND="serial --speed=38400 --unit=0 --word=8 --parity=no --stop=1"`
  - Tell init to start a serial console
    - `sT0:23:respawn:/sbin/getty -L ttyS0 9600 vt100`

SSH Mode: way to go (with network)

- Configure the network
  - Use non-dhcp eth0 in domU
  - Assign in the /etc/hosts of dom0 the names of the domU
  - Assign in the /etc/hosts of domU the name of the dom0

- Configure ssh and access
  - Install openssh-server on the domU
  - Allow x forwarding in the sshd conf
  - Generate ssh keys in dom0
  - Copy public key of the dom0 root in domU
  - Make a alias “ssh=ssh –X” in dom0

PV/HVM Dual Mode Images

- Same IMG can be used in PV and HVM boots
  - A PV generated image is hard to use in a HVM domU
    - Create a partition table
    - Chroot to the img and install grub (using loop-mount)
  - A HVM generated image is easy to deploy as PV
    - An HVM installation can be used straightforward in PV (or KVM)

- How to get a PV ready HVM image?
  - Add hvc0 console in /etc/inittab
    - **`echo "hvc0:2345:respawn:/sbin/getty 38400 hvc0" >>/etc/inittab`**
  - Make sure that disk device name correspond.
    - Usually *hda* in disk line and *xvda1* in append name

PV drivers in HVM domU

- HVM Drivers are very slow
  - QEMU emulated
  - How to run native OS?

- Use PV drivers
  - If kernel supports it, we can para-virtualize selectively device drivers
  - DomU will be able to talk with devices without QEMU intervention
  - Similar to vmWareTools

- Enable PV drivers
  - **`xen_platform_pci=1 in cfg`**
  - Check with
    - **`dmesg | egrep -i 'xen|front'`**
  - We should see how the QEMU device is unplugged and re-plugged at the PV driver
  - Check network and disk:
    - **`ethool -i eth0`** → should return vif driver (8139cp in non-PV)
    - **`cat /proc/partitions`** → should return xvdXX ids (sdaxxx in non-PV)
  - Xen 4.1. **by default** uses PV drivers in 3.xx kernels!!!
  - There are PV drivers for Windows (up to 7 or 2008 Server), Solaris, older Linux, etc…

How to improve even more I/O?

- Bypassing completely VMM
  - Attaching directly the “physical” device to the domU?
  - Preventing other domU/dom0 accessing the reserved device
- Virtualizable devices
  - Also it is possible to split a “physical” device in multiple virtual devices and attach each device to each domU
  - VT-c for NI
  - Generalized for any PCIe and bit restructed to x86 ISA
    - PCI-SIG Single/Multiple Root I/O Virtualization (M/SR-IOV) supported by many non-NI devices (accelerators, GPUs, …)
- Steps
  - Disconnect the device from dom0
  - Connect the device to the domU
  - Use it

PCI-pass-through (or PCI-forwarding)

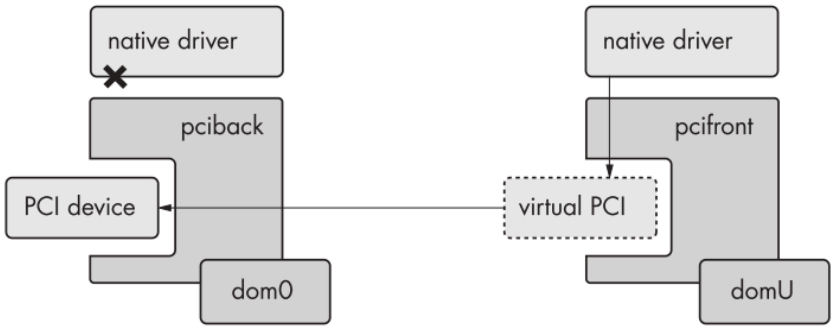
- We need to inspect the pci to identify the device

```
root@aos:~# lspci | grep Etherne
02:01.0 Ethernet controller: Intel Corporation 82545EM Gigabit Ethernet Controller (Copper) (rev 01)
02:05.0 Ethernet controller: Intel Corporation 82545EM Gigabit Ethernet Controller (Copper) (rev 01
```

1. Disconnect the device from dom0 Kernel dynamically

```
modprobe xen-pciback; update-initramfs -u
BDF=0000:02.01.0
#Disconnect device from dom0
echo -n $BDF > /sys/bus/pci/devices/$BDF/driver/unbind
# Add a new slot to the PCI Backend's list
echo -n $BDF > /sys/bus/pci/drivers/pciback/new_slot
# Now that the backend is watching for the slot, bind to it
echo -n $BDF > /sys/bus/pci/drivers/pciback/bind
```

2. Reattach the device to the domU
   - **In domU cfg file**
     - **pci =['02:05.0']**
   - **After asign it, inspect devices**
     - **xm pci-list <domain>**
     - **Device**
     - **0000:02:05.0**



- Requires VT-d to work properly (iommu=soft is somewhat weak).
  - **xm dmesg | grep I\/O**

PV & HVM: Spectrum

	HVM guest	Classic PV on HVM (3.0)	Enhanced PV on HVM (4.0)	PV guest	PVH
Boot sequence	emulated	emulated	emulated	paravirtualized	paravirtualized
Memory	hardware	hardware	hardware	paravirtualized	hardware
Interrupts	emulated	emulated	paravirtualized	paravirtualized	paravirtualized
Timers	emulated	emulated	paravirtualized	paravirtualized	paravirtualized
Disk	emulated	paravirtualized	paravirtualized	paravirtualized	paravirtualized
Network	emulated	paravirtualized	paravirtualized	paravirtualized	paravirtualized
Privilegied operations	hardware	hardware	hardware	paravirtualized	hardware

PVH

- Take advantage of hardware features with **zero** software emulation
- Work in progress mode
  - Available in Xen 4.4, rearchiected in 4.7+
  - Requires 3.14+ linux kernel
- To install Xen 4.4.1 and linux 3.16
  - `#LINUX`
  - `echo "deb http://http.debian.net/debian wheezy-backports main" >> /etc/apt/sources.list`
  - `apt-get update`
  - `apt-get install linux-image-3.16-0.bpo.2-amd64`
  - `#XEN 4.4`
  - `apt-get install libpixman-1-dev bzip2 libaio1 libaio-dev make python-dev gettext bin86 bcc iasl uuid-dev libncurses5-dev pkg-config libglib2.0-dev libyajl-dev libc6-dev-i386 g++ gcc`
  - `wget http://www.xenproject.org/downloads/xen-archives/xen-44-series/xen-441/304-xen-441/file.html -o xen4.4.tar.gz`
  - `tar xf xen4.4.tar.gz`
  - `cd xen-4.4.1`
  - `./configure && make -j 8 world &&` *make xen && make tools && make stuborn*
  - *cd dist/install*
  - *cp -R * / (WARNING!!! DON'T ALLOW TWO INSTALS)*
  - `update-grub`
  - `reboot`

domU config

- Start with a PV config file (assuming pyGRUB)
  - "pvh=1"
  - Uses "xl"
  - Requires full hardware acceleration (processor and chipset)
    - `xl dmesg| grep PVH`
- Why so much effort here?
  - Smaller attack surface than others (smaller interfaces exposed to the software)
  - Boot is a "pain", especially with tens of domUs in the same server
  - Should end providing better performance too…

Xen Status

- Beyond server virtualization
  - XCP
  - SDN (openFlow, vSwitch,…)
- Desktop virtualization
  - BYOD
  - Frame buffer PCI-PT
  - Intel DRM (Direct Rendering Manager)
- Mobile & Embedded
  - Xen on ARM is quite mature
  - Xen on Android? iOS or Android? Android on iOS?...
    - Xen mostly support server or development boards (not phones) due to privative blobs…
    - See https://schd.ws/hosted_files/xensummit2016/eb/slides.pdf