Xen GPU 虛擬叢集建置報告：安裝與測試步驟

1st edition, Sep 2009

涂哲源

格網技術組

國家高速網路與計算中心

Table of Contents

Chapter 1
Introduction

虛擬化可以說是推動雲端服務的核心技術之一，透過虛擬化的技術不僅可以增加設備的彈性與使用率，對於以節能減碳爲訴求的 Green Computing 亦可提供相當大的助益 。虛擬化技術藉由將底層的硬體抽像化處理成共享的硬體資源來達成設備的最大使用效益，使得許多高速計算相關應用得以有效率被管理與執行。

但是，以上情形目前都僅大多數適用在 General Purpose Computing 等相關運用上，相較於最近興起的 GPU(圖形處理器)運算技術來說，虛擬化在圖形處理器上的應用仍是未解的封印；由於各圖形處理器廠商都不太願意公開或是開源他們的驅動程式架構與原始碼，且他們並不熱衷於圖形處理器在虛擬化技術上的應用；因此 GPU Computing 在虛擬化相關技術上的探討與研究並不普遍，況且由於平台的眾多差異性以及硬體間的存取限制等不相容情況下使得相關技術並沒有被正視。

本研究乃是透過 Open Source 的虛擬化解決方案 – Xen 與圖形處理器製造商 NVIDIA 生產的顯示卡來設計實驗平台。從 Fig. 1. 的架構圖可以明顯看出，本實驗平台分別針對高 GPU 運算量以及低 GPU 運算量來提供兩種不同特性的解決方案；在針對高度 GPU 運算的需求上，我們藉由向 Linux kernel 隱藏顯示卡會佔據的 PCI physical address 或是開機完成後再手動移除該顯示卡所佔 PCI slot 的位址聯結等的 Xen PCI Passthrough 方法來處理虛擬化技術在硬體重複定址存取限制上的問題。此方法論也可稱爲是 ” split-driver model”的概念，由 Xen Dom0 作爲 management domain 來處理各類存取、定址與資源分配等問題，而原 NVIDIA module 則須卸載下來並透過 PCI Passthrough 的方式將之隱藏爲 GPU backend module 來抽象化原本在底層的圖形處理單元；而所有在 Xen DomU 上的應用程式如欲存取 GPU 運算資源則需透過每個 Xen DomU 上的 GPU frontend module 來傳送至 management domain 上的 GPU backend module 來分配資源應付相關計算需求。由於可以直接定址並使用原 PCI slot 上的硬體資源，因此較適合運用在具高度 GPU 運算量的案例上。如圖 Fig. 2. 所示即爲 GPU 虛擬化的 stack diagram。

另外，在低 GPU 運算量上我們則是採用 VMGL 的作法來實現，VMGL 是一個開放原始碼的 OpenGL 虛擬化技術解決方案且具備跨平台的特性；目前最新的版本是支援 OpenGL 1.5，因此對於大部分不太複雜的 rendering job 則有不錯的效能與支援。如圖 Fig. 3. 所示即爲 VMGL 的架構圖。藉由透過 GLX transport 的方式來達成 remote rendering，在 remote 端執行的 GL commands 透過此方式傳送至 host 端的 VMGL stub 來轉換爲 direct rendering context 的格式，並直接存取

GPU 的運算資源來處理 remote 的 rendering job 後將畫面輸出。此解決方案亦有提供部分 OpenGL 函數的 suspend 與 resume 功能

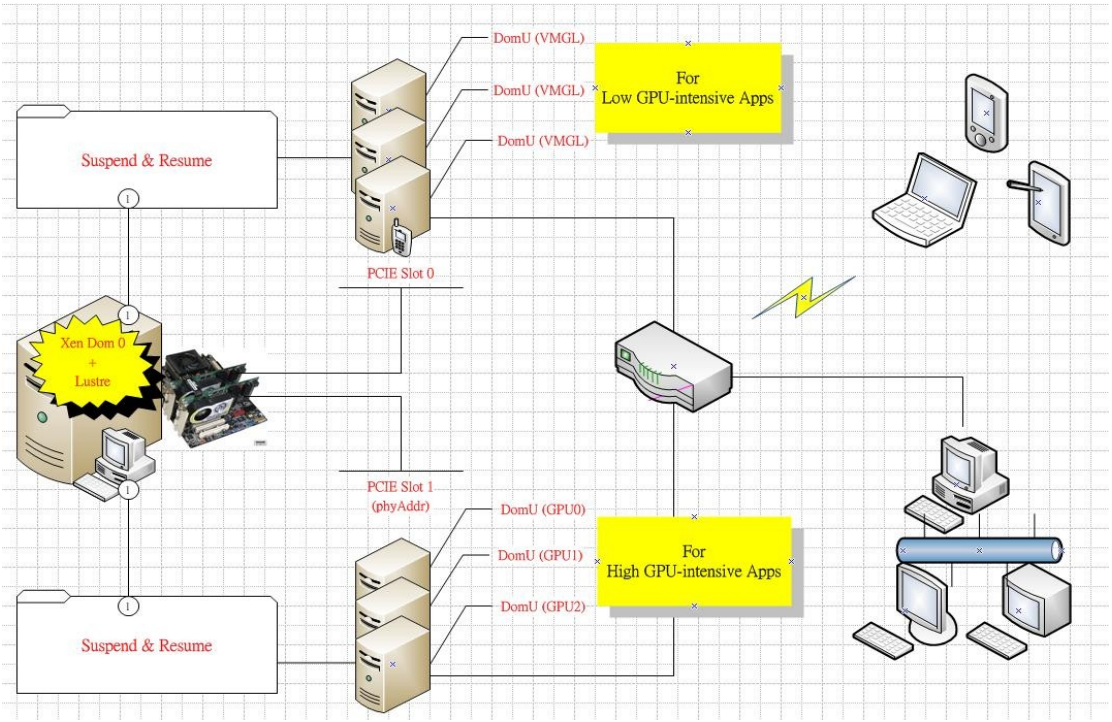

Fig. 1. Xen GPU architecture

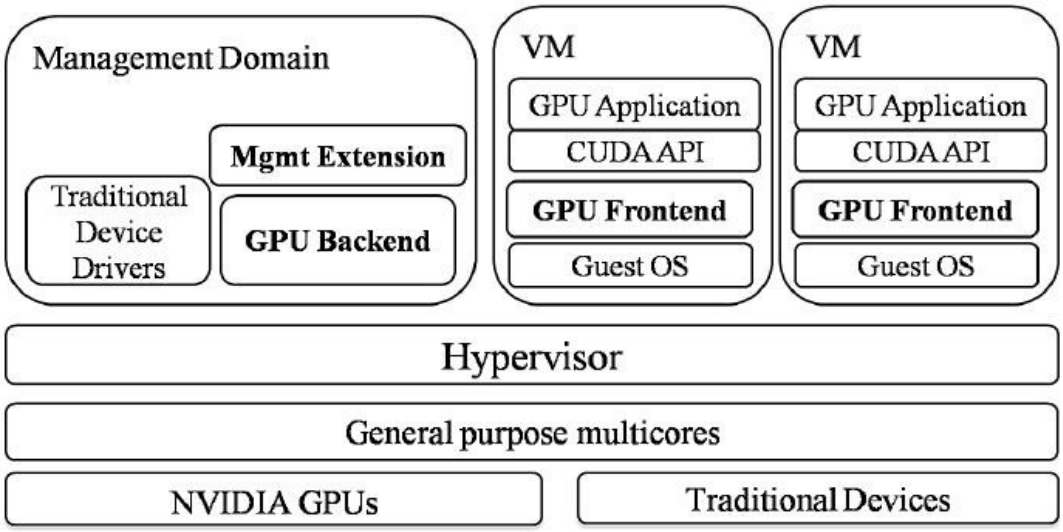

Fig. 2. Virtualization of GPUs (Georgia Tech,2009)

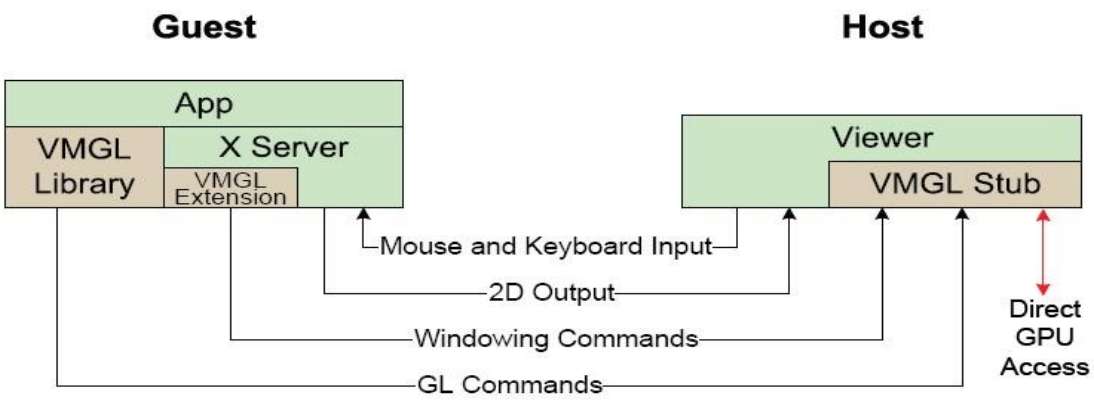

Fig. 3. VMGL Architecture (University of Toronto,2007)

Chapter 2

NVIDIA GPU on XEN

Hardware & Software specifications

Hardware	Software
Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz	Ubuntu 8.10 x86_64
6GB RAM	Kernel 2.6.28
160GB Hard Drive	Xen 3.3.1
NVIDIA GeForce 9800GT 1GB	Lustre 1.6.5.1

Part I 如何將 Dom0 上的顯示卡資源分配給 DomU

Step1: 連線到 Xen Server

rider@cloud:~$ ssh 140.xxx.xxx.xxx

Step2: 先產生一台虛擬機器來使用 CUDA

 # 設定你想要怎樣規格的虛擬機器.

rider@cloud:~$ sudo vim /etc/xen-tools/xen-tools.conf

```
dir = /home
install-method = debootstrap
size      = 8Gb           # Disk image size.
memory = 1024Mb       # Memory size
swap      = 128Mb       # Swap size
fs          = ext3          # use the EXT3 filesystem for the disk image.
dist      = hardy       # Default distribution to install. ---> For CUDA Support (Ubuntu
8.0.4)
image    = sparse      # Specify sparse vs. full disk images.
gateway      = 140.XXX.XXX.XXX
netmask      = 255.255.255.0
broadcast = 140.XXX.XXX.XXX
kernel           = /boot/vmlinuz-`uname -r`
initrd           = /boot/initrd.img-`uname -r`
mirror = http://gb.archive.ubuntu.com/ubuntu/
ext3_options      = noatime,nodiratime,errors=remount-ro
ext2_options      = noatime,nodiratime,errors=remount-ro
xfs_options       = defaults
reiser_options = defaults
```

rider@cloud:~$ sudo xen-create-image --hostname nvidia --ip 140.XXX.XXX.XXX

Step3: 查看你的顯卡資訊

```
rider@cloud:~$ lspci –vv
```

```
01:00.0 VGA compatible controller: nVidia Corporation GeForce 9800 GT
(rev a2)
        Subsystem: ASUSTeK Computer Inc. Device 82a0
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop-
ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort-
<TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Interrupt: pin A routed to IRQ 16
        Region 0: Memory at fd000000 (32-bit, non-prefetchable)
[size=16M]
        Region 1: Memory at d0000000 (64-bit, prefetchable) [size=256M]
        Region 3: Memory at fa000000 (64-bit, non-prefetchable)
[size=32M]
        Region 5: I/O ports at dc80 [size=128]
        [virtual] Expansion ROM at fea00000 [disabled] [size=128K]
        Capabilities: <access denied>
        Kernel driver in use: pciback
    Kernel modules: nvidia, nvidiafb
```

Step4: PCI Frontend Configuration 設定你的 DomU

rider@cloud:~$ sudo vim /etc/xen/nvidia.cfg

```
kernel      = '/boot/vmlinuz-2.6.22.9'
ramdisk     = '/boot/initrd.img-2.6.22.9'

memory      = '1024'
vcpus       = '4'

# 配置你的 PCIE 顯示卡
pci         = ['01:00.0']

root        = '/dev/sda2 ro'
disk        = [
                  'file:/home/domains/nvidia/disk.img,sda2,w',
                  'file:/home/domains/nvidia/swap.img,sda1,w',
```

```
              ]
name        = 'nvidia'

#
#  Networking
#
vif         = [ 'ip=140.xxx.xxx.xxx,mac=00:16:3E:AA:70:5C' ]

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

Step5: PCI Backend Configuration 設定你的 Dom0

rider@cloud:~$ sudo su - w

 # Hide the device from dom0 so pciback can take control.

root@cloud:~$ echo -n "0000:01:00.0" > /sys/bus/pci/drivers/nvidia/unbind

 # Give the dev_ids to pciback, and give it a new slot then bind.

root@cloud:~$ echo -n "0000:01:00.0" > /sys/bus/pci/drivers/pciback/new_slot

root@cloud:~$ echo -n "0000:01:00.0" > /sys/bus/pci/drivers/pciback/bind

root@cloud:~$ cat /sys/bus/pci/drivers/pciback/slots

```
0000:01:00.0
```

 # Caution: Make sure that the device is not controlled by any driver: there should be no driver symlink for nvidia.

```
PATH: /sys/bus/pci/devices/0000:01:00.0/
driver -> ../../../../bus/pci/drivers/nvidia --->This symlink shouldn't exist.
```

Step6: 硬體直接存取設定

Permissive Flag.

rider@cloud:~$ sudo vim /etc/xen/xend-pci-permissive.sxp

```
(unconstrained_dev_ids
      #('0123:4567:89AB:CDEF')
      ('0000:01:00.0')
)
```

User-space Quirks.

rider@cloud:~$ sudo vim /etc/xen/xend-pci-quirks.sxp

```
(pci_ids
   # Entries are formated as follows:
   #      <vendor>:<device>[:<subvendor>:<subdevice>]

   ('10de:0605'    # NVIDIA 9800GT
   )
)
```

Step7: 啓動並登入你的虛擬機器 DomU

說明: 用 root 免密碼先登入,然後建立自己的帳號. 改用自己的帳號登入(亦可用 root 登入 , 不新建帳號):

@ Dom0

rider@cloud:~$ sudo xm create -c nvidia.cfg

@ DomU

root@nvidia:~# adduser username

root@nvidia:~# vim /etc/sudoers

```
username     ALL=(ALL) ALL
```

Step8: 設定你的 DomU 基本環境

 # 設定 locales (系統語系)

rider@nvidia:~$ sudo vim /etc/profile

```
# Locale
export LANGUAGE="en_US.UTF-8"
export LC_ALL="en_US.UTF-8"
export LANG="en_US.UTF-8"
```

rider@nvidia:~$ source /etc/profile

rider@nvidia:~$ sudo dpkg-reconfigure locales

 # 更新 PCI ID DataSheet

rider@nvidia:~$ sudo apt-get update

rider@nvidia:~$ sudo apt-get install wget

rider@nvidia:~$ sudo update-pciids

 # 查看顯卡資訊有無正常顯示

rider@nvidia:~$ lspci

```
00:00.0 VGA compatible controller: nVidia Corporation GeForce 9800 GT (rev a2)
```

 # 查看顯卡資源有無順利分配到 DomU

rider@nvidia:~$ dmesg | grep pci

```
pcifront pci-0: Installing PCI frontend
pcifront pci-0: Creating PCI Frontend Bus 0000:00
pciback 0000:00:00.0: probing...
pciback: pcistub_init_devices_late
```

Chapter 3

NVIDIA CUDA running on XEN

Part Ⅱ 在虛擬機器(Dom0 / DomU)上試跑 CUDA

Step1: 安裝 CUDA Toolkit & SDK 所需套件

 # 環境所需套件

~$ sudo apt-get install autoconf automake build-essential gcc make mesa-common-dev libglu1-mesa-dev mesa-utils libxmu-headers libxmu6 libxmu-dev zlib1g-dev libjpeg62 libjpeg62-dev xutils-dev libxaw-headers libxaw7 libxaw7-dev libxext6 libxext-dev rxvt lwm xauth xvfb xfonts-100dpi xfonts-75dpi culmus xfonts-scalable xfonts-base libtool initramfs-tools libxi6 libxi-dev linux-kernel-devel xserver-xorg xserver-xorg-core xserver-xorg-dev

Step2: 下載 NVIDIA CUDA toolkit

rider@nvidia:~$ mkdir -p nvidia
rider@nvidia:~$ mkdir -p ./nvidia/cuda
rider@nvidia:~$ cd ./nvidia/cuda/
rider@nvidia:~/nvidia/cuda$ wget
http://developer.download.nvidia.com/compute/cuda/2_1/toolkit/cudatoolkit_2.1_linux64_ubuntu8.04.run

Step3: 下載 NVIDIA CUDA SDK

rider@nvidia:~/nvidia/cuda$ wget
http://developer.download.nvidia.com/compute/cuda/2_1/SDK/cuda-sdk-linux-2.10.1215.2015-3233425.run
rider@nvidia:~/nvidia/cuda$ chmod a+x *
rock@cloud:~$ sudo apt-get install autoconf automake build-essential gcc make libtool initramfs-tools libxi6 libxi-dev libxmu6 libxmu-dev xserver-xorg-core xserver-xorg-dev

Step4: 安裝 NVIDIA CUDA toolkit (CUDA Compiler)

rider@nvidia:~/nvidia/cuda$ sudo sh cudatoolkit_2.1_linux64_ubuntu8.04.run

Enter install path (default /usr/local/cuda, '/cuda' will be appended):

…….

* Please make sure your PATH includes /usr/local/cuda/bin
* Please make sure your LD_LIBRARY_PATH includes /usr/local/cuda/lib

* or add /usr/local/cuda/lib to /etc/ld.so.conf and run ldconfig as root

* Please read the release notes in /usr/local/cuda/doc/

* To uninstall CUDA, delete /usr/local/cuda
* Installation Complete

Step5: 安裝 NVIDIA CUDA SDK (CUDA Runtime Libraries)

rider@cloud:~/nvidia/cuda$ sudo sh cuda-sdk-linux-2.10.1215.2015-3233425.run

Enter install path (default /usr/local/cuda, '/cuda' will be appended):
/usr/local/NVIDIA_CUDA_SDK
……
Configuring SDK Makefile
(/usr/local/NVIDIA_CUDA_SDK/common/common.mk)...
* Please make sure your PATH includes /usr/local/cuda/bin
* Please make sure your LD_LIBRARY_PATH includes /usr/local/cuda/lib

* To uninstall the NVIDIA CUDA SDK, please delete
/usr/local/NVIDIA_CUDA_SDK

Step6: 設定 CUDA 執行環境

rider@cloud:~$ sudo su
root@cloud:~$ echo "export PATH=$PATH:/usr/local/cuda/bin" >> /etc/profile
root@cloud:~$ echo "export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib" >> /etc/profile
root@cloud:~$ source /etc/profile
root@cloud:~$ echo "/usr/local/cuda/lib" >> /etc/ld.so.conf
root@cloud:~$ ldconfig
root@cloud:~$ exit

Step7: 換成 gcc-4.1 來編譯

rider@nvidia:~$ sudo apt-get install gcc-4.1 g++-4.1
rider@nvidia:~$ sudo rm /usr/bin/gcc
rider@nvidia:~$ sudo ln -sf /usr/bin/gcc-4.1 /usr/bin/gcc

Step8: Linking CUDA libraries

rider@nvidia:~$ sudo ln -sf /usr/lib/libcuda.so.1 /usr/lib/libcuda.so
rider@nvidia:~$ sudo ln -sf /usr/lib/libglut.so.3 /usr/lib/libglut.so

Step9: 進入 CUDA 專案目錄編譯實驗範例

rider@nvidia:~$ cd /usr/local/NVIDIA_CUDA_SDK/
rider@nvidia:/usr/local/NVIDIA_CUDA_SDK/$ sudo make
rider@nvidia:/usr/local/NVIDIA_CUDA_SDK/projects/deviceQuery$
cd ../../bin/linux/release/

Step10: For example 執行 DeviceQuery 範例

rider@nvidia:/usr/local/NVIDIA_CUDA_SDK/bin/linux/release$ sudo ./deviceQuery

There is 1 device supporting CUDA

Device 0: "GeForce 9800 GT"

Major revision number:	1
Minor revision number:	1
Total amount of global memory:	1073414144 bytes
Number of multiprocessors:	14
Number of cores:	112
Total amount of constant memory:	65536 bytes
Total amount of shared memory per block:	16384 bytes
Total number of registers available per block: 8192	
Warp size:	32
Maximum number of threads per block:	512
Maximum sizes of each dimension of a block:	512 x 512 x 64
Maximum sizes of each dimension of a grid:	65535 x 65535 x 1
Maximum memory pitch:	262144 bytes
Texture alignment:	256 bytes

Clock rate: 1.51 GHz
Concurrent copy and execution: Yes

Test PASSED

Press ENTER to exit...w

Chapter 4
Virtual Machine Graphics Acceleration

Part Ⅲ 在虛擬機器上獲得 GPU 硬體加速
- 使用 VMGL

Step1: Dom0 上安裝 VMGL 所需套件

 # 安裝環境所需套件.

~$ sudo apt-get install build-essential firefox flashplugin-nonfree htop libao2 libasound2 libmikmod2 mesa-common-dev libglu1-mesa-dev mesa-utils libxmu-headers libxmu6 libxmu-dev zlib1g-dev libjpeg62 libjpeg62-dev xutils-dev libxext6 libxext-dev rxvt lwm tightvncserver xauth xvfb xfonts-100dpi xfonts-75dpi culmus xfonts-scalable xfonts-base

Step2: 下載 VMGL.

rider@cloud:~/vmgl$ wget
http://www.cs.toronto.edu/~andreslc/software/vmgl-0.1.tar.bz2
rider@cloud:~/vmgl$ tar jxvf vmgl-0.1.tar.bz2
rider@cloud:~/vmgl$ cd ./vmgl.hg/tightvnc/

Step3: VMGL patch for AMD64_Machine.

rider@cloud:~/vmgl$ wget --no-check-certificate
https://trac.nchc.org.tw/grid/raw-attachment/wiki/Xen_GPU_cluster/tightvnc-1.2.9-amd64support.patch
rider@cloud:~/vmgl/vmgl.hg/tightvnc$ patch -p0
< ../../tightvnc-1.2.9-amd64support.patch
rider@cloud:~/vmgl$ cd ..

Step4: 編譯與安裝

rider@cloud:~/vmgl$ sudo ln -sf /usr/bin/make /usr/bin/gmake
rider@cloud:~/vmgl/vmgl.hg$ make
rider@cloud:~/vmgl/vmgl.hg$ sudo make install-host

Step5: 設定主機信任清單. 在 Dom0 加入 DomU

rider@cloud:~$ xauth

Set the authority for remote guest.

Using authority file /home/rider/.Xauthority
xauth> add guest/unix:10 MIT-MAGIC-COOKIE-1
ec0ffd387888b9749d55f88031505888 -> guest 端的 VMGL DISPLAY:10
xauth> add guest/unix:1 MIT-MAGIC-COOKIE-1
6824789b4ce0ac5743aeb57fd3ef8f5b -> guest 端的 VNC DISPLAY:1
xauth> exit

---

rider@cloud:~$ xauth list

---

guest:1 MIT-MAGIC-COOKIE-1 5c53c8c640b816d7714e23d5cfcd9a8e
cloud/unix:1 MIT-MAGIC-COOKIE-1 5c53c8c640b816d7714e23d5cfcd9a8e
guest:2 MIT-MAGIC-COOKIE-1 5bfe43007be14ca93e9ee5fc71701463
cloud/unix:2 MIT-MAGIC-COOKIE-1 5bfe43007be14ca93e9ee5fc71701463
guest:3 MIT-MAGIC-COOKIE-1 04499062b48a199921e859ee76d267ab
cloud/unix:3 MIT-MAGIC-COOKIE-1 04499062b48a199921e859ee76d267ab
guest/unix:10 MIT-MAGIC-COOKIE-1 ec0ffd387888b9749d55f88031505888

```
guest/unix:1    MIT-MAGIC-COOKIE-1    6824789b4ce0ac5743aeb57fd3ef8f5b
guest/unix:2    MIT-MAGIC-COOKIE-1    98e85e7e551246d428023b54b5dff02b
cloud/unix:0    MIT-MAGIC-COOKIE-1    b5f3b4b4f9d0e63e8f9b88a6d57fad15
localhost:0    MIT-MAGIC-COOKIE-1    b5f3b4b4f9d0e63e8f9b88a6d57fad15
```

Step6: 建構一台 VMGL Guest VM

 # 設定你想要怎樣規格的虛擬機器.

```
rider@cloud:~$ sudo vim /etc/xen-tools/xen-tools.conf
```

```
dir = /home
install-method = debootstrap
size      = 8Gb           # Disk image size.
memory = 1024Mb       # Memory size
swap      = 128Mb       # Swap size
fs          = ext3          # use the EXT3 filesystem for the disk image.
dist      = hardy       # Default distribution to install. ---> For CUDA Support (Ubuntu
8.0.4)
image    = sparse      # Specify sparse vs. full disk images.
gateway      = 140.XXX.XXX.XXX
netmask      = 255.255.255.0
broadcast = 140.XXX.XXX.XXX
kernel           = /boot/vmlinuz-`uname -r`
initrd           = /boot/initrd.img-`uname -r`
mirror = http://free.nchc.org.tw/ubuntu/
ext3_options      = noatime,nodiratime,errors=remount-ro
ext2_options      = noatime,nodiratime,errors=remount-ro
xfs_options       = defaults
reiser_options = defaults
```

```
rider@cloud:~$ sudo xen-create-image --hostname guest --ip 140.XXX.XXX.XXX
```

Step7: 設定虛擬機器(Guest)

```
rider@cloud:~$ sudo vim /etc/xen/nvidia.cfg
```

```
kernel          = '/boot/vmlinuz-2.6.22.9'
ramdisk         = '/boot/initrd.img-2.6.22.9'
```

```
memory          = '1024'
vcpus           = '4'

# 不需配置 PCIE 顯示卡
# pci             = ['01:00.0']

root            = '/dev/sda2 ro'
disk            = [
                        'file:/home/domains/nvidia/disk.img,sda2,w',
                        'file:/home/domains/nvidia/swap.img,sda1,w',
                    ]
name            = 'nvidia'

#
#   Networking
#
vif             = [ 'ip=140.xxx.xxx.xxx,mac=00:16:3E:AA:70:5C' ]

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

Step8: 在 DomU(Guest) 上安裝 VMGL

 # 啓動虛擬機器 Guest 並登入.

 # 方法一.

```
rider@cloud:~$ sudo xm create -c guest.cfg
```

 # 方法二.

```
@Dom0
rider@cloud:~$ sudo xm create guest.cfg
@ClientUser
rider@PC:~$ ssh 140.xxx.xxx.xxx
```

 # 安裝環境所需套件.

~$ sudo apt-get install build-essential firefox flashplugin-nonfree htop mozilla-plugin-vlc libao2 libasound2 libmikmod2 mesa-common-dev libglu1-mesa-dev mesa-utils libxmu-headers libxmu6 libxmu-dev zlib1g-dev libjpeg62 libjpeg62-dev xutils-dev libxaw-headers libxaw7 libxaw7-dev libxext6 libxext-dev rxvt lwm vlc vlc-plugin-alsa tightvncserver xauth xvfb xfonts-100dpi xfonts-75dpi culmus xfonts-scalable xfonts-base

 # 基本桌面環境.

Desgin for lightweight X support

xfonts-base xfonts-100dpi xfonts-75dpi -> fonts support
rxvt -> x-terminal-emulator for VNC
lwm -> x-window-manager for VNC

Step9: 下載 VMGL.

rider@guest:~$ mkdir vmgl
rider@guest:~$ cd ./vmgl
rider@guest:~/vmgl$ wget
http://www.cs.toronto.edu/~andreslc/software/vmgl-0.1.tar.bz2
rider@guest:~/vmgl$ wget
http://trac.nchc.org.tw/grid/raw-attachment/wiki/Xen_GPU_cluster/tightvnc-1.2.9-amd64support.patch
rider@guest:~/vmgl$ tar jxvf vmgl-0.1.tar.bz2
rider@guest:~/vmgl$ cd ./vmgl.hg/tightvnc/

Step10: VMGL patch for AMD64_Machine.

rider@cloud:~/vmgl$ wget --no-check-certificate
https://trac.nchc.org.tw/grid/raw-attachment/wiki/Xen_GPU_cluster/tightvnc-1.2.9-amd64support.patch
rider@cloud:~/vmgl/vmgl.hg/tightvnc$ patch -p0
< ../../tightvnc-1.2.9-amd64support.patch
rider@cloud:~/vmgl$ cd ..

Step11: 編譯與安裝

rider@guest:~/vmgl/vmgl.hg$ sudo mkdir -p /usr/lib/xorg/modules/extensions
rider@cloud:~/vmgl$ sudo ln -sf /usr/bin/make /usr/bin/gmake
rider@cloud:~/vmgl/vmgl.hg$ make
rider@cloud:~/vmgl/vmgl.hg$ sudo make install-guest

 # 修正 rgb PATH 問題.

Fix the rgb_Path problem.
rider@guest:~/vmgl/vmgl.hg$ sudo mkdir /usr/X11R6/lib/X11
rider@guest:~/vmgl/vmgl.hg$ sudo ln -sf /etc/X11/rgb.txt /usr/X11R6/lib/X11/rgb

---

Step12: 在 DomU(Guest) 上安裝輕量級桌面環境(LXDE)

rider@client:~$ sudo su
 # 加入 LXDE 的 source.
root@client:/home/rider# echo "deb http://ppa.launchpad.net/lxde/ubuntu hardy main" >> /etc/apt/sources.list
root@client:/home/rider# echo "deb-src http://ppa.launchpad.net/lxde/ubuntu hardy main" >> /etc/apt/sources.list
root@client:/home/rider# exit
 # 安裝 LXDE .
rider@client:~$ sudo apt-get install lxde openbox-themes gnome-settings-daemon gnome-settings-daemon-dev gnome-icon-theme
 # 預設 LXDE 爲 VNC 桌面環境 .
rider@client:~$ echo "startlxde &" >> .vnc/xstartup
 # 基本桌面環境.

Desgin for lightweight X support

lxde -> Lightweight X11 Desktop Environment
openbox -> X-window-manager for VNC

Step13: VMGL 使用說明: 在 Dom0 上的設定

 # 將 X-Windows 桌面 export 出來.

rider@cloud:~$ export DISPLAY=:0

 # 啓動 VLGL stub-daemon.

rider@cloud:~$ stub-daemon

 # 檢查 stub-daemon 所對應的 port(7000) 有無開啓.

rider@cloud:~$ netstat –tunlp

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

PID/Program name

tcp 0 0 0.0.0.0:8002 0.0.0.0:*

LISTEN -

tcp 0 0 0.0.0.0:22 0.0.0.0:*

LISTEN -

tcp 0 0 127.0.0.1:631 0.0.0.0:*

LISTEN -

tcp 0 0 0.0.0.0:7000 0.0.0.0:*

LISTEN 29082/stub-daemon ---> VMGL stub-daemon

udp 0 0 0.0.0.0:32769 0.0.0.0:*

-

udp 0 0 0.0.0.0:5353 0.0.0.0:*

-

Step14: VMGL 使用說明: 在 DomU 上的設定

 # 設定 DomU 上的字型路徑對應.

rider@guest:~$ sudo ln -sf /usr/share/fonts/X11/ /usr/X11R6/lib/X11/fonts (if necessary - fix font path)

FontPath:

/usr/X11R6/lib/X11/fonts

or

/usr/share/fonts/X11

 # 確認 rgb 存在於對應路徑.

rider@guest:~$ less /usr/X11R6/lib/X11/rgb.txt (if necessary - rgb path confirmation)

 # 設定 DomU 上的 VMGL 環境變數.

rider@guest:~$ sudo vim /etc/profile

GLSTUB=Cloud_IP(host_IP):7000
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/vmgl
LD_PRELOAD=/usr/local/lib/vmgl/libGL.so
export GLSTUB LD_LIBRARY_PATH LD_PRELOAD

---

rider@guest:~$ source /etc/profile

 # 設定 DomU 上的 X forwarding.

Using X forwarding

rider@guest:~$ sudo vim /etc/ssh/sshd_config.

X11Forwarding yes

---

rider@guest:~$ sudo vim /etc/ssh/ssh_config

ForwardX11 yes
ForwardX11Trusted yes

 # 重新啓動 DomU 上的 SSH Service.

rider@guest:~$ sudo /etc/init.d/ssh restart

 # 設定 VNC server 使用者密碼.

rider@guest:~$ vncpasswd

~$ vncserver → Activate VNC server
xauth: creating new authority file /home/gtd/.Xauthority

New 'X' desktop is guest:1

Creating default startup script /home/gtd/.vnc/xstartup
Starting applications specified in /home/gtd/.vnc/xstartup
Log file is /home/gtd/.vnc/guest:1.log

 # 設定 VNC server 啓動參數.

```
rider@guest:~$ vim .vnc/xstartup
```

```
# xrdb $HOME/.Xresources
```

 # 開啓 DomU 上的第一台 VNC Server.

```
rider@guest:~$ vncserver -geometry 1024x768 -depth 24 :1
```

 # 檢查 VNCserver 有無開啓對應之 port(5901 開始).

```
rider@guest:~$ netstat –tunlp
```

```
(Not all processes could be identified, non-owned process info
  will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
PID/Program name
tcp        0      0 0.0.0.0:5901                0.0.0.0:*
LISTEN      10652/Xtightvnc ---> guest:1
tcp        0      0 0.0.0.0:5902                0.0.0.0:*
LISTEN      10630/Xtightvnc
tcp        0      0 0.0.0.0:6001                0.0.0.0:*
LISTEN      10652/Xtightvnc ---> guest:2
tcp        0      0 0.0.0.0:6002                0.0.0.0:*
LISTEN      10630/Xtightvnc
tcp        0      0 127.0.0.1:6010              0.0.0.0:*
LISTEN      -
tcp6       0      0 :::22                        :::*
LISTEN      -
tcp6       0      0 ::1:6010                     :::*
LISTEN      -
```

 # 將 VNC Server:1 export 出來.

```
rider@guest:~$ export DISPLAY=:1
```

Step15: 在 Dom0 上 透過 X forwarding 登入 DomU

 # 使用 X forwarding 的方式登入 DomU.

Using X forwarding

rider@cloud:~$ ssh -X guest

 # 檢查 direct rendering 有無起來.

rider@guest:~$ glxinfo

---

name of display: localhost:10.0
display: localhost:10 screen: 0
direct rendering: Yes
server glx vendor string: VMGL
server glx version string: 1.2 VMGL
server glx extensions:
client glx vendor string: VMGL
client glx version string: 1.2 VMGL
client glx extensions:
GLX version: 1.3
GLX extensions:
OpenGL extensions:
    GL_ARB_depth_texture, GL_ARB_fragment_program, GL_ARB_multisample,
    GL_ARB_multitexture, GL_ARB_occlusion_query,
GL_ARB_point_parameters,
    GL_ARB_point_sprite, GL_ARB_shadow, GL_ARB_texture_border_clamp,
    GL_ARB_texture_compression, GL_ARB_texture_cube_map,
    GL_ARB_texture_env_add, GL_ARB_texture_env_combine,
    GL_EXT_texture_env_combine, GL_ARB_texture_env_dot3,
    GL_EXT_texture_env_dot3, GL_ARB_texture_mirrored_repeat,
    GL_ARB_texture_non_power_of_two, GL_ARB_transpose_matrix,
    GL_ARB_vertex_buffer_object, GL_ARB_vertex_program,
GL_ARB_window_pos,
    GL_EXT_blend_color, GL_EXT_blend_minmax,
GL_EXT_blend_func_separate,
    GL_EXT_blend_subtract, GL_EXT_texture_env_add, GL_EXT_fog_coord,
    GL_EXT_multi_draw_arrays, GL_EXT_secondary_color,
GL_EXT_shadow_funcs,
    GL_EXT_stencil_wrap, GL_EXT_texture_cube_map,
GL_EXT_texture_edge_clamp,
    GL_EXT_texture_filter_anisotropic, GL_EXT_texture_lod_bias,
    GL_EXT_texture_object, GL_EXT_texture3D, GL_EXT_bgra,
    GL_IBM_rasterpos_clip, GL_NV_fog_distance, GL_NV_fragment_program,
    GL_NV_register_combiners, GL_NV_register_combiners2,

GL_NV_texgen_reflection, GL_NV_texture_rectangle,
GL_NV_vertex_program,
GL_NV_vertex_program1_1, GL_NV_vertex_program2,
GL_SGIS_generate_mipmap,
GL_CR_state_parameter, GL_CR_cursor_position, GL_CR_bounding_box,
GL_CR_print_string, GL_CR_tilesort_info, GL_CR_synchronization,
GL_CR_head_spu_name, GL_CR_performance_info, GL_CR_window_size,
GL_CR_tile_info, GL_CR_saveframe, GL_CR_readback_barrier_size,
GL_CR_server_id_sharing, GL_CR_server_matrix
GLX_ARB_multisample
OpenGL vendor string: H. Andres Lagar-Cavilla
OpenGL renderer string: VMGL
OpenGL version string: 1.5 VMGL 1.9
0x52 24 dc 0 32 0 r y . 8 8 8 0 4 24 8 16 16 16 16 2 1 None
0x53 24 dc 0 32 0 r y . 8 8 8 8 4 24 8 16 16 16 16 2 1 None
0x54 24 dc 0 32 0 r y . 8 8 8 0 4 24 8 16 16 16 16 4 1 None
0x55 24 dc 0 32 0 r y . 8 8 8 8 4 24 8 16 16 16 16 4 1 None
0x56 24 dc 0 32 0 r . . 8 8 8 0 4 24 8 16 16 16 16 2 1 None
0x57 24 dc 0 32 0 r . . 8 8 8 8 4 24 8 16 16 16 16 2 1 None
0x58 24 dc 0 32 0 r . . 8 8 8 0 4 24 8 16 16 16 16 4 1 None
0x59 24 dc 0 32 0 r . . 8 8 8 8 4 24 8 16 16 16 16 4 1 None
0x23 32 tc 0 32 0 r y . 8 8 8 0 4 24 8 16 16 16 16 0 0 None
0x5a 32 tc 0 32 0 r y . 8 8 8 8 4 24 8 16 16 16 16 0 0 None
0x5b 32 tc 0 32 0 r . . 8 8 8 0 4 24 8 16 16 16 16 0 0 None
0x5c 32 tc 0 32 0 r . . 8 8 8 8 4 24 8 16 16 16 16 0 0 None
0x5d 32 tc 0 32 0 r y . 8 8 8 0 4 24 0 16 16 16 16 0 0 None
0x5e 32 tc 0 32 0 r y . 8 8 8 8 4 24 0 16 16 16 16 0 0 None
0x5f 32 tc 0 32 0 r . . 8 8 8 0 4 24 0 16 16 16 16 0 0 None
0x60 32 tc 0 32 0 r . . 8 8 8 8 4 24 0 16 16 16 16 0 0 None
0x61 32 tc 0 32 0 r y . 8 8 8 0 4 0 0 16 16 16 16 0 0 None
0x62 32 tc 0 32 0 r y . 8 8 8 8 4 0 0 16 16 16 16 0 0 None
0x63 32 tc 0 32 0 r . . 8 8 8 0 4 0 0 16 16 16 16 0 0 None
0x64 32 tc 0 32 0 r . . 8 8 8 8 4 0 0 16 16 16 16 0 0 None
0x65 32 tc 0 32 0 r y . 8 8 8 0 4 24 0 16 16 16 16 2 1 None
0x66 32 tc 0 32 0 r y . 8 8 8 8 4 24 0 16 16 16 16 2 1 None
0x67 32 tc 0 32 0 r y . 8 8 8 0 4 24 0 16 16 16 16 4 1 None
0x68 32 tc 0 32 0 r y . 8 8 8 8 4 24 0 16 16 16 16 4 1 None
0x69 32 tc 0 32 0 r . . 8 8 8 0 4 24 0 16 16 16 16 2 1 None

```
0x6a 32 tc  0 32  0 r  .  .  8  8  8  8  4 24  0 16 16 16 16  2 1 None
0x6b 32 tc  0 32  0 r  .  .  8  8  8  0  4 24  0 16 16 16 16  4 1 None
0x6c 32 tc  0 32  0 r  .  .  8  8  8  8  4 24  0 16 16 16 16  4 1 None
0x6d 32 tc  0 32  0 r  y  .  8  8  8  0  4 24  8 16 16 16 16  2 1 None
0x6e 32 tc  0 32  0 r  y  .  8  8  8  8  4 24  8 16 16 16 16  2 1 None
0x6f 32 tc  0 32  0 r  y  .  8  8  8  0  4 24  8 16 16 16 16  4 1 None
0x70 32 tc  0 32  0 r  y  .  8  8  8  8  4 24  8 16 16 16 16  4 1 None
0x71 32 tc  0 32  0 r  .  .  8  8  8  0  4 24  8 16 16 16 16  2 1 None
0x72 32 tc  0 32  0 r  .  .  8  8  8  8  4 24  8 16 16 16 16  2 1 None
0x73 32 tc  0 32  0 r  .  .  8  8  8  0  4 24  8 16 16 16 16  4 1 None
0x74 32 tc  0 32  0 r  .  .  8  8  8  8  4 24  8 16 16 16 16  4 1 None
```

 # DomU 上的 glxgears 效能測試.

rider@guest:~$ glxgears

```
47819 frames in 5.0 seconds = 9563.678 FPS
46064 frames in 5.0 seconds = 9212.566 FPS
44584 frames in 5.0 seconds = 8916.581 FPS
44256 frames in 5.0 seconds = 8850.974 FPS
44688 frames in 5.0 seconds = 8937.528 FPS
```

Step16: 使用 VNC Viewer 登入 DomU

 # 透過 VNC Viewer(TurboVNC Viewer or Tightvnc viewer)登入 DomU 的 VNC Server,登入後便可以透過 rxvt 來下指令跑相關 3D 應用了.

@ ClientUser

rider@PC:~$ vncviewer guest:1

 # 檢查 direct rendering=yes?.

@ VNC viewer 登入 :1 後

rider@guest:~$ glxinfo

Chapter 5

Suspension & Resumption

Part IV Suspension & Resumption for your 3D-Apps

Part1: Between DomU

實驗一: Pausing & Unpausing

 # 當使用者正在使用虛擬機器看影片,玩 Game 或是執行其他 3D-Apps 時. 使用者在暫停目前的工作後,便交由"Dom0"來作 snapshot.

@ Dom0

rider@cloud:~$ sudo xm list

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	3851	4	r-----	40036.9
client	6	1024	4	-b----	21.9
guest	7	1024	4	-b----	13.6

 # 將虛擬機器 "guest" (DomainID:8) 暫停.

rider@cloud:~$ sudo xm pause 8

 # 檢查虛擬機器狀態,虛擬機器 "guest" 已經 "pause".

rider@cloud:~$ sudo xm list

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	3851	4	r-----	40059.8
client	6	1024	4	-b----	38.9
guest	8	1024	4	--p---	3.0

 # 重新將虛擬機器 "guest" (DomainID:8) 還原回原工作狀態.

rider@cloud:~$ sudo xm unpause 8

 # 檢查虛擬機器狀態.

rider@cloud:~$ sudo xm list

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	3851	4	r-----	40036.9
client	6	1024	4	-b----	21.9
guest	8	1024	4	r-----	13.6

 # 使用者可以再重新透過 VNC Viewer 連回去虛擬機器 "guest" 繼續原本的工作.

@ ClientUser

rider@PC:~$ vncviewer guest:1

說明: # "DomU" 被暫停後會將目前狀態儲存在記憶體,一旦 "Dom0" 重開機後該 "DomU" 的狀態將會消失.

實驗二: Saving & Restoring

 # 當使用者正在使用虛擬機器看影片,玩 Game 或是執行其他 3D-Apps 時. 使用者在暫停目前的工作後,便交由"Dom0"來作 snapshot.

@ Dom0

rider@cloud:~$ sudo xm list

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	3851	4	r-----	40036.9
client	6	1024	4	-b----	21.9
guest	7	1024	4	-b----	13.6

 # 將虛擬機器 "guest" 作 snapshot.

rider@cloud:~$ sudo xm save -c guest ./backup/snapshot/guest.chk

 # 將虛擬機器 "guest" 暫時關閉並釋放資源.

rider@cloud:~$ sudo xm destroy guest

 # 重新將虛擬機器 "guest" 還原回原工作狀態.

rider@cloud:~$ rider@cloud:~$ sudo xm restore ./backup/snapshot/guest.chk

 # 使用者可以再重新透過 VNC Viewer 連回去虛擬機器 "guest" 繼續原本的工作.

@ ClientUser

rider@PC:~$ vncviewer guest:1

說明: # **"DomU" 被 snapshot 後會將執行狀態存成一各檔案並存放在硬碟中指定目錄,就算 "Dom0" 重開機後還是可以將 "DomU" 狀態還原回來.**

實驗三: Suspension & Resumption

 # 必須先使用 xm new 來新增虛擬機器給 Dom0, 再使用 xm start 來啓動虛擬機器. 直接用 xm create 會有問題 ?!
 # 當使用者正在使用虛擬機器看影片,玩 Game 或是執行其他 3D-Apps 時. 使用者可以在任何時刻中止目前的工作,亦可隨時隨地還原狀態.
@ DomU
 # 啓動 Suspend & Resume daemon, 並指向 Dom0 stub-daemon port.
rider@guest:~$ cd ./vmgl/vmgl.hg/extra/suspend-resume-helper
rider@guest:~/vmgl/vmgl.hg/extra/suspend-resume-helper$ make
rider@guest:~/vmgl/vmgl.hg/extra/suspend-resume-helper$ ./sr_daemon Dom0_IP:7000 &
rider@guest:~/vmgl/vmgl.hg/extra/suspend-resume-helper$ ps aux | grep sr_daemon

rider	3326	0.0	0.1	34056	1352 pts/0	S	15:56	0:00 ./sr_daemon Dom0_IP:7000

@ Dom0
rider@cloud:~$ sudo xm list

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	3724	4	r-----	80.5
client	4	1024	4	-b----	19.0
guest	5	1024	4	-b----	0.3

 # 將虛擬機器 "guest" suspend (中止).
rider@cloud:~$ sudo xm suspend guest

 # 查看虛擬機狀態可以發現 "guest" 已經被 suspend.

rider@cloud:~$ sudo xm list

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	3724	4	r-----	86.3
client	4	1024	4	-b----	20.6
guest		1024	4		3.8

 # 將虛擬機器 "guest" resume (恢復) 回原來狀態 .

rider@cloud:~$ sudo xm resume guest

 # 查看虛擬機狀態可以發現 "guest" 已經 resume 回來.

rider@cloud:~$ sudo xm list

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	3724	4	r-----	92.0
client	4	1024	4	-b----	23.8
guest	6	1024	4	-b----	0.3

Reference

1. NVIDIA CUDA: http://www.nvidia.com/object/cuda_home.html
2. openSUSE NVIDIA + Xen: http://en.opensuse.org/Use_Nvidia_driver_with_Xen
3. NVIDIA GPUs DEV_IDs: http://www.laptopvideo2go.com/forum/index.php?showtopic=7664
4. pci_ids db: http://www.pcidatabase.com/
5. Xen: assigning PCI devices to a domain: http://www.bestgrid.org/index.php/Xen:_assigning_PCI_devices_to_a_domain
6. Xen PCI Passthrough: http://www.wlug.org.nz/XenPciPassthrough
7. Xen Users' Manual v3.0: http://www.cl.cam.ac.uk/research/srg/netos/xen/readmes/user/
8. VMGL support: https://code.fluendo.com/elisa/trac/wiki
9. Latest NVIDIA driver: ftp://download.nvidia.com/XFree86/Linux-x86_64/
10. GViM: http://www.cc.gatech.edu/~vishakha/files/GViM.pdf
11. VMM-Independent Graphics Acceleration: http://andres.lagarcavilla.com/publications/LagarCavillaVEE07.pdf