2014 National Tsing Hua University Summer Exchange

(LI Xiaodong, Shandong University, Computer Science, X1023063)

Software-based GPU Virtualization

Keyword: KVM, VDI, Pass-through, vGPU, Iptable Forwarding, Cloud Computing

Advisor: Prof. Yeh-Ching Chung, Ph.D. Yu-Shiang Lin

Abstract In September of 2013, NVidia Grid K1 achieved GPU hardware virtualization for the first time. However, it needs special environments supported. With the hardware upgrading, a Software-based vGPU solution is welcomed by high performance cloud providers. This report tells the research process and advanced theories of GPU virtualization in the world, and introduces our work on KVM. As a necessary instrument, various kinds of desktop virtualization are researched. The report is divided into seven parts according to the meeting time, including a conclusion of the two month exchange research. The timetable is as below.

◆ 07.07~07.10	Virtualization Techniques Classification and Installation
◆ 07.11~07.20	KVM on CentOS and NVidia Drivers
◆ 07.21~07.30	NVidia GPU and GPU Virtualization
◆ 07.31~08.05	Desktop Virtualization Research
◆ 08.06~08.13	Quadro6000 Pass-through
◆ 08.14~08.21	Spice VDI and IPtable Forwarding

I. Virtualization Techniques

Though having been put forward for decades, It is still difficult to give a convincing definition of cloud computing. Since 2008, IBM describes cloud computing as a delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility over a network. [1] Through this description, a flexible delivery is needed. Rather than the tradition opinion that one machine one computer, the cloud computing needs to virtualize its hardware to take full use of the workstation. In this part, three questions are solved as below.

1. What is virtualization technique;
2. The differences between popular virtualization techniques;
3. How to achieve creation of virtual machines, and let these them can be connected with each other and communicate with outside network domain.

By IBM definition, virtualization technique is that by hiding specific physical characteristics of computing platform, users can be provided with abstract and unified simulation computing environment (virtual machine). The main purpose is to efficiently using mainframes' resources and achieves isolation.

Virtualization technique has many different classification methods. The classic classification method is

Just as Fig.1 describing below.

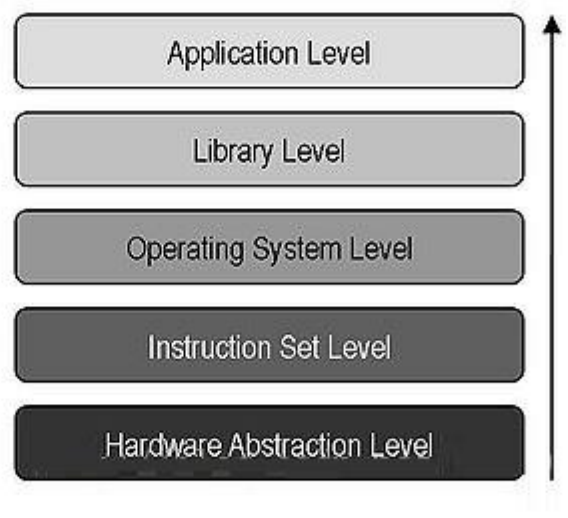


Fig.1 Different levels of virtualization techniques

In addition, there are two popular classification methods of hypervisors or virtual machine manager (VMM). One method is as below.

- ◆ Type 1 (or native, bare metal) hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems. A guest operating-system thus runs on another level above the hypervisor.
  - Oracle VM Server for SPARC, x86
  - The Citrix XenServer
  - VMware ESX/ESXi
  - Microsoft Hyper-V 2008/2012.
- ◆ Type 2 (or hosted) hypervisors run within a conventional operating-system environment. With the hypervisor layer as a distinct second software level, guest operating-systems run at the third level above the hardware.
  - VMware Workstation
  - VirtualBox

Another classification method can be described in the aspect of guest programs.

- ◆ Full Virtualization: complete simulation, guest operating system unmodified.
- ◆ Partial Virtualization: Some target environment is simulated, so some guest programs need modifications to run in this virtual environment.
- ◆ Para-virtualization: A hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment.

KVM, QEMU, Xen and VMware are four most popular hardware virtualization techniques in the world.

QEMU is a generic and open source machine emulator and virtualizer. When used as a machine emulator, QEMU can run OS and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). By using dynamic translation, it achieves very good performance. When used as a virtualizer, QEMU achieves near native performances by executing the guest code directly on the host CPU. QEMU supports virtualization when executing under the Xen hypervisor or using the KVM kernel module in Linux. When using KVM, QEMU can virtualize x86, server and embedded PowerPC, and S390 guests.

Xen™ is a virtual machine monitor for x86 that supports execution of multiple guest operating systems with unprecedented levels of performance and resource isolation. Xen is Open Source software, released under the terms of the GNU General Public License. Xen is a hypervisor using a microkernel design, providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently.

VMware, Inc. is an American software company that provides cloud and virtualization software and services, and was the first to successfully virtualize the x86 architecture. VMware developed a range of products, most notable of which are their hypervisors. VMware software provides a completely virtualized set of hardware to the guest operating system. VMware's products predate the virtualization extensions to the x86 instruction set, and do not require virtualization-enabled processors. Although VMware virtual machines run in user-mode, VMware Workstation itself requires the installation of various drivers in the host operating-system.

The main difference between these techniques can be described as below.

Tab． 1 Difference between KVM, QEMU, Xen and VMware

Name	Host CPU	Guest CPU	Host OS	Guest OS
KVM	Intel/AMD CPU with x86 Virtualization	x86/x86-64	Linux	Linux，Windows
QEMU	x86，x86-64，IA-64，PowerPC，Alpha，SPARC 32 and 64，ARM，S/390，M68k	x86，x86-64，ARM，SPARC 32 and 64，PowerPC，MIPS	Windows，Linux，Mac OS X，Solaris，FreeBSD，OpenBSD，BeOS	Changed
QEMU（kqemu）	Intel x86，x86-64	Intel x86，x86-64	Linux，FreeBSD，OpenBSD，Solaris，Windows	Changed
QEMU（qvm86）	x86	x86	Linux，NetBSD，Windows	Changed
VMware ESX Server	x86, x86-64	x86, x86-64	Bare installation	Windows，Red Hat，SuSE，Ubuntu，Netware，Solaris，FreeBSD
VMware ESXi Server	x86，x86-64	x86，x86-64	Bare installation	Windows，Red Hat，SuSE，Ubuntu，Netware，Solaris，FreeBSD
VMware Fusion	x86，Intel VT-x	x86，x86-64	Mac OS X（Intel）	Windows，Linux，Netware，Solaris
VMware Server	x86，x86-64	x86，x86-64	Windows，Linux	DOS，Windows，Linux，FreeBSD，Netware，Solaris
VMware Workstation 6.0	x86，x86-64	x86，x86-64	Windows，Linux	DOS，Windows，Linux，FreeBSD，Netware，Solaris，Darwin

VMware Player 2.0	x86，x86-64	x86，x86-64	Windows，Linux	DOS，Windows，Linux，FreeBSD，Netware，Solaris，Darwin
Xen	x86，x86-64	x86，x86-64	NetBSD，Linux，Solaris	Linux, Solaris, Windows XP & 2003 Server, FreeBSD

Actually, KVM is more complicated in the cloud computing world.

- ◆ KVM are implemented as a kernel module for Linux. When loaded, KVM allows its host operating system to act as a bare-metal (i.e., Type 1) hypervisor.
- ◆ Linux distributions are operating systems in their own right, so KVM also can be regarded as Type 2 hypervisors.

Therefore, there are companies that call these kind of specific hypervisor implementations like KVM and bhyve Type 0 (Zero) Hypervisor. However, there is no consensus so far.

These virtualization classifications can be descried as below.

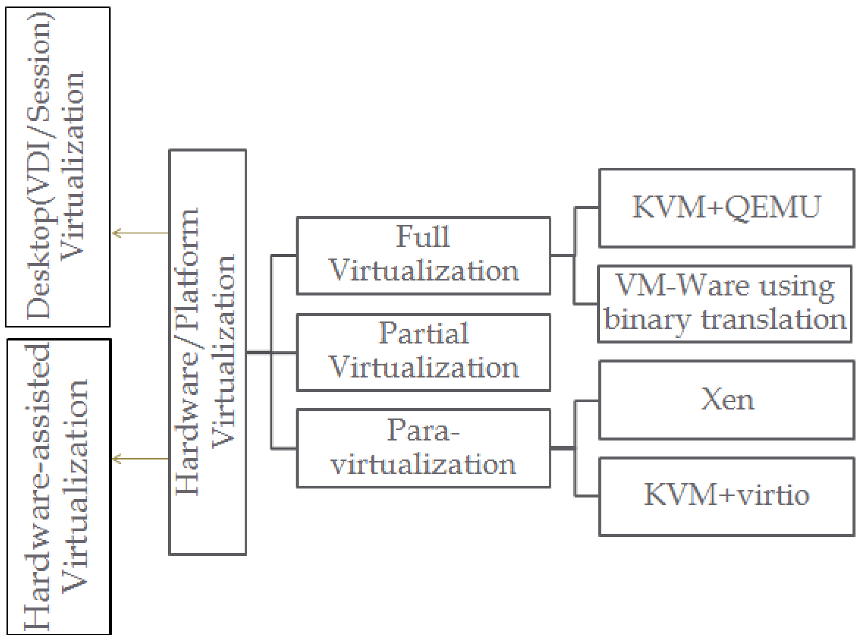

Desktop(VDI/Session) Virtualization
Hardware-assisted Virtualization
Hardware/Platform Virtualization
Full Virtualization
Partial Virtualization
Para-virtualization
KVM+QEMU
VM-Ware using binary translation
Xen
KVM+virtio

Fig.2 virtualization classifications

The first installation is KVM on Ubuntu. The report briefly records the history of operations on Ubuntu as below.

```
$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils
$ sudo adduser `id -un` libvirtd
  (Relogin)
$ virsh -c qemu:///system list
$ sudo apt-get install virt-manager
```

NAT is used to make connection among VMs and outside network domain. These steps can be concluded as process as below.

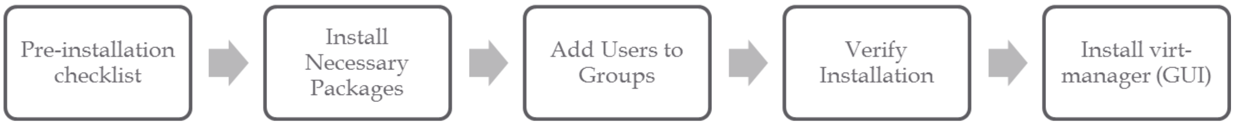

Pre-installation checklist
Install Necessary Packages
Add Users to Groups
Verify Installation
Install virt-manager (GUI)

Fig.3 KVM on Ubuntu Installation Process

II. KVM on CentOS and NVidia Drivers

In order to coordinate the work of the whole lab, the host operating system is changed to CentOS 6.5 server mini and all results of this report below are in CentOS environment. The configuration of CentOS has some slight difference compared with Ubuntu but this stable OS helps a lot. There are some attentions in the process of Centos6.5 server min installation.

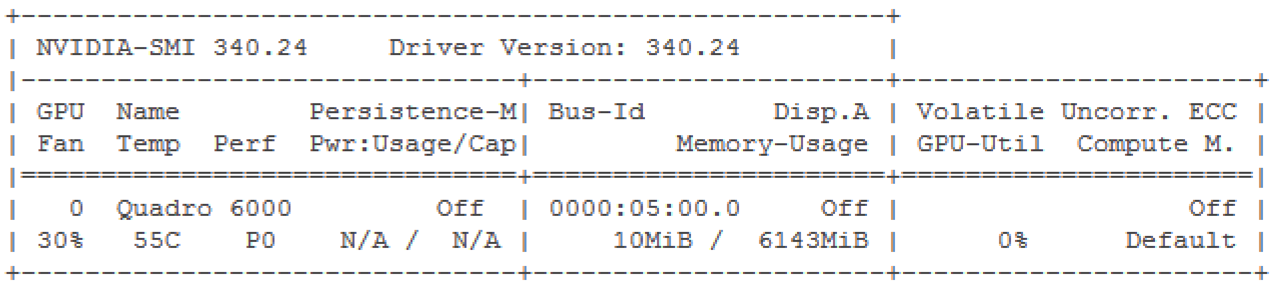
- The installation on the server will meet an problem (no matter CD or USB, CD is easier):
  Detecting hardware…
  Solution: "Tab" + "acpi=off"
- Network: /etc/sysconfig/network + /etc/resolv.conf + /etc/sysconfig/network-scripts/ifcfg-eth0
  Then, use SSH to control the server with NVidia Quadro 6000.

Next, the pre-installation process of NVidia driver is as below.

# yum groupinstall "Development Tools"
# yum install kernel-devel kernel-headers dkms
# /sbin/lspci | grep NVIDIA
(to show the information of the hardware, then find the proper driver: LINUX X64 (AMD64/EM64T) DISPLAY DRIVER)

Installation:

1. open "*/etc/modprobe.d/blacklist.conf" and add "blacklist nouveau*"
2. create a new "initramfs" file and taking backup of existing:
   # mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.bak
   # dracut -v /boot/initramfs-$(uname -r).img $(uname -r)
3. *# sudo yum update + reboot*
4. *chmod +x NVIDIA-Linux-x86_64-340.24.run + ./NVIDIA-Linux-x86_64-340.24.run*
5. testing: *nvidia-smi*



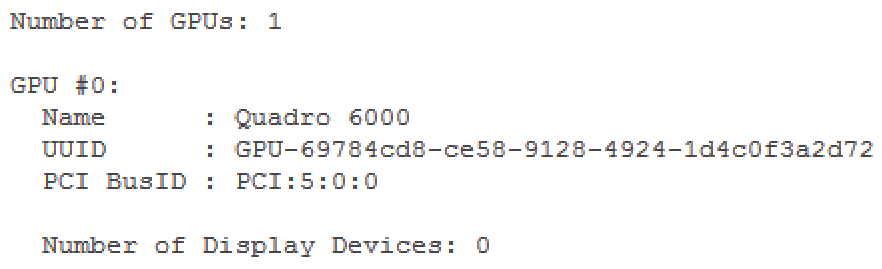
```
+------------------------------------------------------+
| NVIDIA-SMI 340.24     Driver Version: 340.24         |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Quadro 6000          Off  | 0000:05:00.0     Off |                  Off |
| 30%   55C    P0    N/A /  N/A |     10MiB /  6143MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

Installation of cuda_6.0:

```
# sudo /usr/bin/nvidia-uninstall
# yum install cuda-repo-rhel6-6.0-37.x86_64.rpm
#wget
http://developer.download.nvidia.com/compute/cuda/6_0/rel/installers/cuda_6.0.37_linux_64.run
# chmod +x cuda_6.0.37_linux_64.run
#./cuda_6.0.37_linux_64.run
# export PATH=/usr/local/cuda-6.0/bin:$PATH
# export LD_LIBRARY_PATH=/usr/local/cuda-6.0/lib64:$LD_LIBRARY_PATH
```

To verify:

```
# nvidia-xconfig -query-gpu-info
```



```
Number of GPUs: 1

GPU #0:
  Name      : Quadro 6000
  UUID      : GPU-69784cd8-ce58-9128-4924-1d4c0f3a2d72
  PCI BusID : PCI:5:0:0

  Number of Display Devices: 0
```

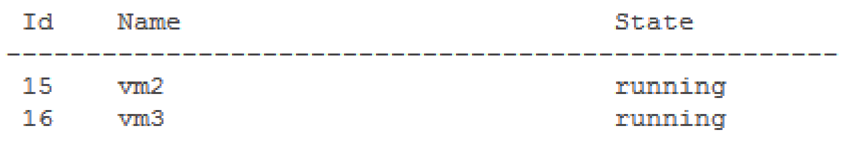
To install KVM:

```
# yum -y install @virt* dejavu-lgc-* xorg-x11-xauth tigervnc \ libguestfs-tools policycoreutils-python bridge-utils
```

Creating VMs on KVM is much easier to use VMs by GUI; however, it is not suitable for server. Using X Server is possible. The server send instructions and the remote OS translate them into GUI. It is the same in turn.

2 steps to use X Server:

1. In remote OS(Windows 7 here):
   install Xming and Xming-fonts(in the same fold), then configure putty
2. In Centos Server:
   yum -y groupinstall "Desktop" "Desktop Platform" "X Window System" "Fonts"



```
 Id    Name                           State
----------------------------------------------------
 15    vm2                            running
 16    vm3                            running
```

By comparison, KVM hypervisor is easy to use, and virtual machines are created successfully. In addition, the acceleration of CPU is checked that the physical machine is suitable (*egrep '(vmx|svm)' --color=always /proc/cpuinfo*). Attentions: using Wlan to connect server is not a good idea, since the session is not stable and usually appear the error: Access Denied and using wired session is more stable; balance the efficiency and the performance that using X Server is easier for use, but using Desktop Platform will obviously decreasing the OS performance. Just using virt-manager on GUI is acceptable.

III. NVidia GPU and GPU Virtualization

Modern graphics co-processors (GPUs) can produce high fidelity images several orders of magnitude faster than general purpose CPUs, and this performance expectation is rapidly becoming ubiquitous in personal computers. Despite this, GPU virtualization is a nascent field of research. In September of 2013, NVidia GRID series GPUs become the first GPUs able to achieve virtualization in the world. However, NVIDIA® GRID™ vGPU™ only support compatible version of Citrix XenServer by now, and too many users are still using last generation GPU such as Quadro 6000 with other kind of hypervisors.

In this report, physics GPU is Quadro 6000, for its excellent ability to the acceleration of 3D work. Here, we can compare it with the GPU specially for vGPU techniques: Grid K1 and K2.

Tab． 2 Differences between Grid and Quadro 6000 GPU

	GRID		**Quadro 6000**
Chief Application	Virtual desktop and Applications with GPU Acceleration		Graphic workstation desktop professional graphics: CUDA parallel computing and 3D rendering
Multi-desktops Achievement	GRID vGPU: GraphicsCodes Pass-through to GPU Directly Without Hypervisor Encoding		NVIDIA® Mosaic
Appearance Time	2013		2011
GPU Virtualization	YES(First to achieve GPU hardware virtualization)		NO
Architecture	NVIDIA Kepler		NVIDIA Fermi
	GRID K1	**GRID K2**	
GPU Number	4 Kepler GPU	2 Advanced Kepler GPU	1 GPU
NVIDIA CUDA Core Number	768	3072	448
Memory Capacity	16 GB DDR3	8 GB GDDR5	6GB GDDR5
Memory Interfaces	128bit	128bit	320bit
PMAX	130 W	225 W	204 W
Length	10.5 inch	10.5 inch	9.75 inch
Height	4.4 inch	4.4 inch	4.376 inch
Breadth	Double Groove	Double Groove	Double Groove
Aux Power	6	8	8 or two of 6
PCIe	Third generation	Third generation	Second generation
Cooling	Passive	Passive	Cooling fan
Attention	1. Only support compatible version of Citrix XenServer support vGPU;		1. Multi-GPU is based on NVIDIA Kepler™
	2. Dedicated GPU only VMware vSphere Hypervisor compatible.		2. Last generation of GPU

GPU virtualization, also called vGPU, is a new technique to accelerate the virtualization process even in cloud computing. From the table above, we can see that special hardware is needed during GPU virtualization. The hypervisor with vGPU Manager only support compatible versions of Citrix XenServer (XenServer 6.2

SP1 & XenDesktop 7.1). With the developing of hardware and hypervisor, it is meaningful to achieve software vGPU without the help of hardware. The features of vGPU can be concluded as below.

- Achieve GPU hardware acceleration when sharing a GPU among multiple users
- Application features and compatibility maintain unmodified
- Graphics commands of each virtual machine are passed directly to the GPU, without translation by the hypervisor(also called Pass-through)

Tab. 3 vGPU Virtualization Degree

NVIDIA GRID Graphics Board	Virtual GPU Profile	Application Certifications	Graphics Memory	Max Displays Per User	Max Resolution Per Display	Max Users Per Graphics Board	Use Case
GRID K2	K260Q	Yes	2,048 MB	4	2560x1600	4	Designer/ Power User
	K240Q	Yes	1,024 MB	2	2560x1600	8	Designer/ Power User
	K220Q	Yes	512 MB	2	2560x1600	16	Designer/ Power User
	K200		256 MB	2	1900x1200	16	Knowledge Worker
GRID K1	K140Q	Yes	1,024 MB	2	2560x1600	16	Power User
	K120Q	Yes	512 MB	2	2560x1600	32	Power User
	K100		256 MB	2	1900x1200	32	Knowledge Worker

vGPU Manager assigns just the right amount of memory to meet each user, that is, eight users share each physical GPU:

*Max user number = 8 * GPU numbers*
(GRID k1=4, GRID k2=2)

Therefore, we can get that
k1: up to 32 VMs on a single board and up to 100 users are supported.

NVidia GRID series GPUs are easy to use in companies that provide services including gaming, application, desktop, infrastructure, platform, or software as a service, most of which are not achieved so far. It is worth paying more attention in this field.

Quadro 6000 also has many characters in virtualization aspect.

- One user per physical GPU (GRID GPU Encoding）
- GPU Pass-Through(special hypervisor needed)
- NVidia Driver is the same, so the applications ensure compatibility.

Actually, as long as supporting GPU pass-through, GPU virtualization is supported. Quadro 6000 is also believed to achieve GPU shared among multi-users, though special hypervisor is needed and vGPU manager need to be written by ourselves at this time.

.

IV. Desktop Virtualization Research

Desktop virtualization is software technology that separates the desktop environment and associated application software from the physical client device that is used to access it. Remote desktop virtualization can also be provided via a Cloud computing similar to that provided using Software as a service model. This approach is usually referred to as Cloud Hosted Virtual Desktops.

These definitions and advantages can be described briefly as below.

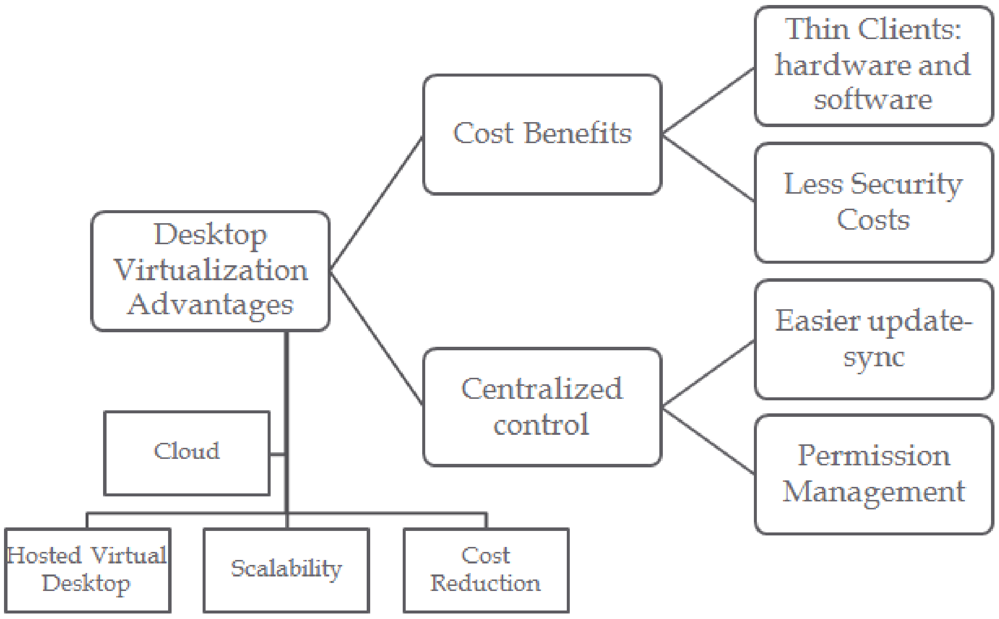


Fig.4 Desktop Virtualization

There are two kind of desktop virtualization:

virtual desktop infrastructure (VDI):

- The host maintains multiple VMs for multiple users
- Users have remote desktops from the same host
- Products: XenDesktop, VDI-in-a-Box, QVD, Microsoft VDI

Session Virtualization:

- Each user has a desktop and a personal folder a shared PC with multiple monitors and keyboards
- A shared PC with multiple monitors and keyboards
- Products: Remote Desktop Services, QVD, (Terminal Services) Chrome

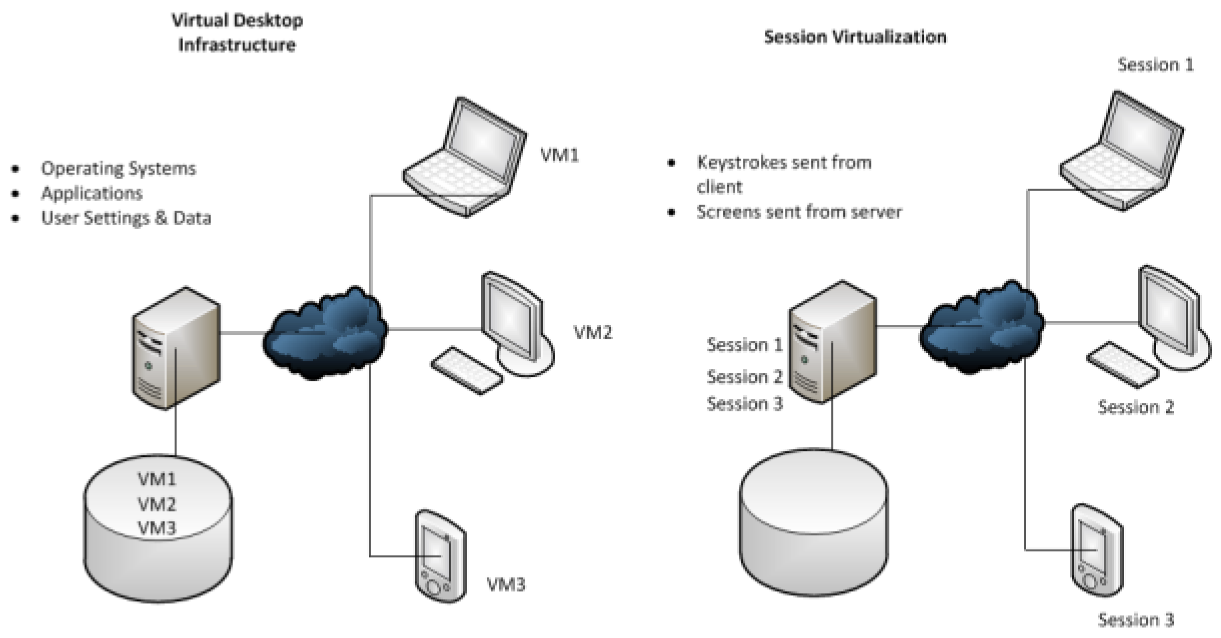


Fig.5 Two kinds of Desktop Virtualizations

To conclude, there are two main differences between VDI and Session virtualization from the side of different kinds of users.

1. *VDI is for users need significant control of their environment, but more expensive*
2. *Session virtualization may be 25% the price of VDI, but less freedom*

During this report, many VDI and session methods have been used, and session virtualization is easier to achieve at this time. However, since VDI gives users more freedom and more isolation, VDI has more development space in the future.

Without doubt, desktop virtualization will reduce more financial investment in the future, especially in organizations using computers centralized such as companies or government. VDI is a popular way to achieve desktop virtualization, but session virtualization is a cheaper one. Comparison of these kinds of desktop virtualization is important, but due to the GPU virtualization calls for many virtual machines, VDI is needed in the report. QVD is used at first, but changed to Spice due to the reason of open source. It will be introduced in part VI.

V. Quadro6000 Pass-through

Device pass-through is the foundation of device sharing among virtual machines. It is the same with GPU virtualization. After GPU pass-through, all things to do are just to achieve GPU table forwarding by hypervisor and fully software-based GPU virtualization is not difficult by then.

Our target is to:

1. *patch the kernel and QEMU for better compatibility with graphics card / VGA VFIO pass-through*
2. *create and configure a new virtual machine (VM) with real hardware(Quadro6000) attached to it*

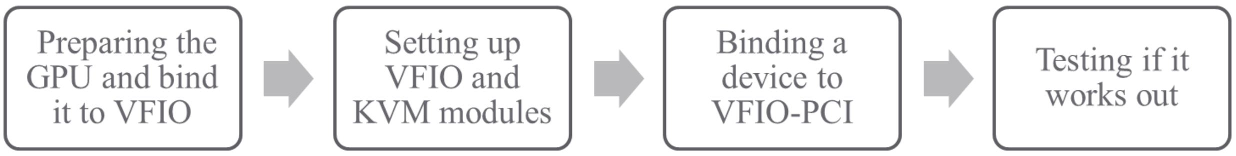

Preparing the GPU and bind it to VFIO
Setting up VFIO and KVM modules
Binding a device to VFIO-PCI
Testing if it works out

Fig.6 Process of Quadro6000 Pass-through

However, the latest NVidia driver (337.88) specifically checks for KVM as the hypervisor, and the error below followed.

ERROR: Unable to build the NVIDIA kernel module

To make use of GPU pass-through with virtual machines running Windows and Linux, the hardware platform must support the following features:

- A CPU with hardware-assisted instruction set virtualization: Intel VT-x or AMD-V.

```
# egrep -c '(vmx|svm)' /proc/cpuinfo
    32
```

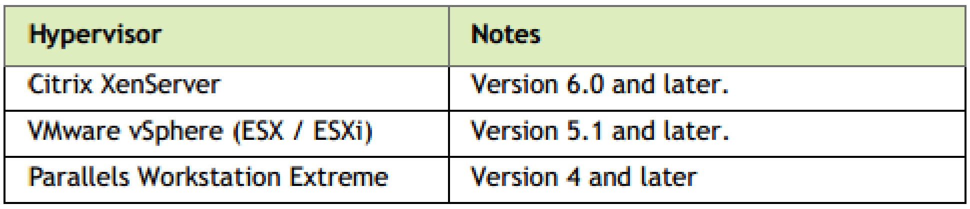
- Platform support for I/O DMA remapping.

On Intel platforms the DMA remapper technology is called Intel VT-d.
On AMD platforms it is called AMD IOMMU.

The information of NVidia Quadro6000 in detail is as below, indicating that the NVidia driver works fine. We checks the AMD IOMMU from BIOS of our host computer for I/O DMA remapping, and everything works fine. Therefore, we started to guess that NVidia on KVM cannot support GPU pass-through, and the discovery of Tab. 4 verify it.

```
05:00.0 Class 0300: Device 10de:06d8 (rev a3)
        Subsystem: Device 10de:076f
        Flags: bus master, fast devsel, latency 0, IRQ 24
        Memory at fc000000 (32-bit, non-prefetchable) [size=32M]
        Memory at d8000000 (64-bit, prefetchable) [size=128M]
        Memory at d4000000 (64-bit, prefetchable) [size=64M]
        I/O ports at e800 [size=128]
        [virtual] Expansion ROM at fbf80000 [disabled] [size=512K]
        Capabilities: [60] Power Management version 3
        Capabilities: [68] MSI: Enable- Count=1/1 Maskable- 64bit+
        Capabilities: [78] Express Endpoint, MSI 00
        Capabilities: [b4] Vendor Specific Information <?>
        Capabilities: [100] Virtual Channel <?>
        Capabilities: [128] Power Budgeting <?>
        Capabilities: [600] Vendor Specific Information <?>
        Kernel driver in use: nvidia
        Kernel modules: nvidia, nouveau, nvidiafb
```

Tab. 4 Quadro6000 Virtualization Supported Hypervisors



Hypervisor	Notes
Citrix XenServer	Version 6.0 and later.
VMware vSphere (ESX / ESXi)	Version 5.1 and later.
Parallels Workstation Extreme	Version 4 and later

Though wasting much time in this problem, we have found the way to achieve NVidia GPU virtualization on KVM, and the action itself is a creation in the world.

Through researching, we believe there are two different potential solutions to this problem.

- Removing or changing the KVM signature is sufficient for the driver to load and work. This only changes the visibility of KVM to the guest and para-virtual features specifically tied to the KVM CPU-ID.
- Patch the NVidia driver to avoid it, but it is not easy due to the lack of source code of NVidia latest dirver.

VI. Spice VDI and IP-table Forwarding

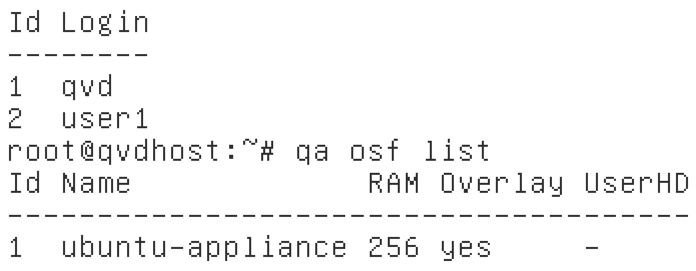
At first, we used QVD as a VDI solution, and built an environment to run the demo-version.

1. *Ubuntu 12.04 (Precise Pangolin) GNU/Linux operating system with Virtual Box*
2. *CPU supporting virtualization and disk allocated at least 8G with*

When tested, it works fine.

1. *Install a QVD client on remote computer*
2. *Add an account in the server and link it with a VM in free*

The result below is a QVD running on the Ubuntu server.



```
Id Login
--------
1  qvd
2  user1
root@qvdhost:~# qa osf list
Id Name             RAM Overlay UserHD
--------------------------------------
1  ubuntu-appliance 256 yes     -
```

However, QVD may not open its source code of the latest version, and the production released is not free. Therefore, we have to turn to Spice.

Spice is a complete open source solution for interaction with virtualized desktop devices. The Spice project deals with both the virtualized devices and the front-end. Interaction between front-end and back-end is done using VD-Interfaces. The VD-Interfaces (VDI) enables both ends of the solution to be easily utilized by a third-party component.

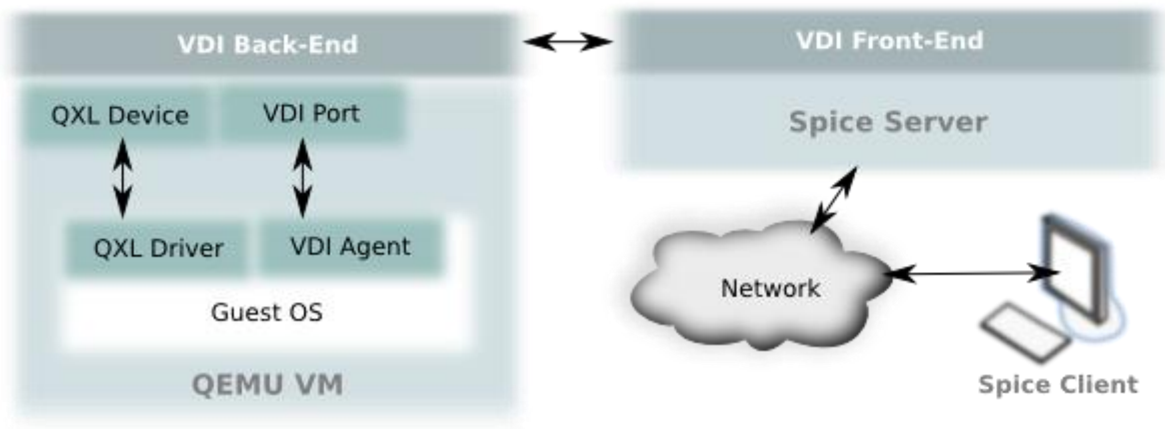


Fig.7 Brief Theory of Spice Solution on Top of QEMU

The Spice project plans to provide additional solutions, including: [3]

1. *Remote access for a physical machine*
2. *VM front-end for local users (i.e., render on and share devices of the same physical machine)*

The installation of Spice on CentOS6.5 Server Mini:

Install RPM(with GUI):

yum install cairo-spice ffmpeg-spice-libs
ngspice pixman-spice spice-glib spice-gtk spice-
gtk-python spice-gtk-tools spice-xpi

Configuration:

Remmove: VNC Display
Add Graphics: Spice Server
Video Model: qxl

Remote Control:

Listen on all Public Interfaces, and choose a password for the VM
(Remember the port number for each Virtual Machine for using in Spice client.)

Spice uses port-forwarding, so each port matches the corresponding virtual machine though the IP address is the same. IP-table Forwarding uses the same theory and the operation has some slight differences.

Client for Windows:	*Windows binaries*
Client for Linux:	*GTK+ widget*
Client for Web Browser:	*spice-xpi*
Client for Android:	*aSpice*

(You also can find Spice client in Chrome Application Store.)

VII. Conclusion

Though just in charge of a small part of Software-based GPU Virtualization, I feel stimulated and take part in the team work with interest. Virtualization Techniques Classification and Installation helps choose KVM on CentOS in the end. The study of NVidia Drivers helps realize the problem of Quadro6000 Pass-through. NVidia GPU and GPU Virtualization make me involved in the core of the project, and Desktop Virtualization Research helps find Spice VDI to become the final form of Software-based GPU Virtualization. IP-table Forwarding helps realize what to do after the success of GPU pass-through to become truly Software-based.

According to Gartner's Hype cycle, cloud computing has reached a maturity that leads it into a productive phase. This means that most of the main issues with cloud computing have been addressed to a degree that clouds have become interesting for full commercial exploitation. This however does not mean that all the problems have actually been solved, only that the according risks can be tolerated to a certain degree. Cloud computing is therefore still as much a research topic, as it is a market offering. In this condition, virtualization including GPU virtualization has a promising research potential. [2]

In the end, thanks to the help of my advisor: Prof. Yeh-Ching Chung, Ph.D. Yu-Shiang Lin, as well as the students in SSLAB, I can finish my work and enjoy the journey both in academy and in my life. I appreciate it more than I can say!

[1] "The NIST Definition of Cloud Computing". National Institute of Standards and Technology. Retrieved 24 July 2011.
[2] Smith, David Mitchell. "Hype Cycle for Cloud Computing, 2013". Gartner. Retrieved 3 July 2014.
[3] Spice Project, REDHAT, “http://www.spice-space.org/”