A Performance Comparison of VMware GPU Virtualization Techniques in Cloud Gaming

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Zhihong Zhuo

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Cloud gaming is an application deployment scenario which runs an interactive gaming application remotely in a cloud according to the commands received from a thin client and streams the scenes as a video sequence back to the client over the Internet, and it is of interest to both research community and industry. The academic community has developed some open-source cloud gaming systems such as GamingAnywhere for research study, while some industrial pioneers such as Onlive and Gaikai have succeeded in gaining a large user base in the cloud gaming market.

Graphical Processing Unit (GPU) virtualization plays an important role in such an environment as it is a critical component that allows virtual machines to run 3D applications with performance guarantees. Currently, GPU pass-through and GPU sharing are the two main techniques of GPU virtualization. The former enables a single virtual machine to access a physical GPU directly and exclusively, while the latter makes a physical GPU shareable by multiple virtual machines. VMware Inc., one of the most popular virtualization solution vendors, has provided concrete implementations of GPU pass-through and GPU sharing. In particular, it provides a GPU pass-through solution called Virtual Dedicated Graphics Acceleration (vDGA) and a GPU-sharing solution called Virtual Shared Graphics Acceleration (vSGA). Moreover, VMware Inc. recently claimed it realized another GPU sharing solution called vGPU. Nevertheless, the feasibility and performance of these solutions in cloud gaming has not been studied yet.

In this work, an experimental study is conducted to evaluate the feasibility and performance of GPU pass-through and GPU sharing solutions offered by VMware in cloud gaming scenarios. The primary results confirm that vDGA and vGPU techniques can fit the demands of cloud gaming. In particular, these two solutions achieved good performance in the tested graphics card benchmarks, and gained acceptable image quality and response delay for the tested games.

Acknowledgements

I would like to take this opportunity to express my gratitude to those people who once helped me and made the successful completion of this thesis. First and foremost, I want to express my sincere appreciation and genuine gratitude to my supervisors Dr. Derek Eager and Dr. Dwight Makaroff who helped me all the way from the beginning of my master program at U of S. When I started my program at U of S, both of my supervisors guided me in the right directions and provided a lots of support to help me resolve all the problems I encountered with in my study. Additionally, they offered a lot of suggestions and ideas when I started looking for my thesis topic. I also appreciate they spent a lot of their valuable time in correcting the mistakes in my thesis. Besides my supervisors, I would like to thank the rest of the members of my thesis committee: Dr. Kevin Stanley and Dr. Ray Spiteri for their suggestions. I am also very thankful to my friends who once granted my unconditional encouragement and support. Last but not the least, I would like to express genuine gratitude to my parents and my sister who were always supportive to me.

Contents

List of Tables

List of Figures

List of Abbreviations

API	Application Programming Interface
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DGA	Dedicated Graphics Acceleration
FPS	First Person Shooter
fps	Frames Per Second
GA	GamingAnywhere
GPGPU	General-purpose Graphics Processing Units
GPU	Graphical Processing Unit
HPC	High Performance Computing
MKS	mouse-keyword-screen
MMU	Memory Management Unit
ND	Network Delay
OD	Playout Delay
OpenCL	Open Computing Language
OpenGL	Open Graphics Library
OS	Operating System
PD	Processing Delay
PSNR	Peak Signal to Noise Ratio
QoE	Quality of Experience
QoS	Quality of Service
RDP	Remote Desktop Protocol
RPG	Role Playing Games
RTS	Real Time Strategy
RTT	Network Round-trip Time
SGA	Shared Graphics Acceleration
SLA	Service-level Agreement
SSIM	Structural Similarity Index
TLB	Translation Lookaside Buffer
vDGA	Virtual Dedicated Graphics Acceleration
VMM	Virtual Machine Monitor
vSGA	Virtual Shared Graphics Acceleration

Chapter 1

Introduction

1.1 Virtualization

Virtualization is a technology which combines or divides computing resources to present one or many operating environments, using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, time-sharing, and others" [33]. A virtualization layer is an essential component in virtualization as it provides the capability of using hardware resources to create multiple virtual machines with isolation and performance guarantees. Sometimes, such a virtualization layer is also called hypervisor or Virtual Machine Monitor (VMM) [33]. A virtual machine is defined as an emulation of a computer that provides environments for supporting the operating system (OS).

Today, most computer resources, particularly Central Processing Unit (CPU), have been able to be well-virtualized with performance guarantees by using software-based virtualization techniques such as full virtualization and para-virtualization [14]. In addition, some hardware features like Intel VT-d [1] and AMD-V[1] were introduced for better virtualization support. Moreover, the virtualization of some hardware such as GPUs, which was a problem before, now is being resolved with a variety of strategies.

Modern CPUs perform multi-taking well, but GPUs have been notably bad at multi-tasking with multiple graphics-intensive applications. In addition, as GPU designers typically were secretive about the specifications regarding the design and implementation of GPU products, there was only limited accessible documentation. Moreover, as GPU architectures varied a lot across generations and the generational cycle was relatively short compared to the CPU and other devices, it was normally unrealistic to specify a virtual device for each modern GPU [14]. These issues, however, now are being solved by industry and the research community as many rich GPU applications are presenting rising demand for full GPU virtualization [14][42]. For instance, recent hardware advances have allowed virtualization systems to do one-to-one mapping between a virtual machine and a physical GPU [14].

GPU sharing, where multiple virtual GPUs share a physical GPU, is emerging as an attractive research area [42]. Not only software-based solutions like device emulation, but hardware techniques are also introduced to realize GPU sharing with the guarantee of higher performance and lower overheads.[2] In particular,

[1] AMD-V Introduction. Website: http://www.amd.com/en-us/solutions/servers/virtualization. Accessed: Oct 12, 2015.
[2] Overview of Virtual GPU Technology. Website: http://www.nvidia.ca/object/virtual-gpus.html. Accessed: Oct 12, 2015.

VMware Inc.,[3] one of the most successful virtualization solution vendors, has developed solutions to support GPU pass-through and GPU sharing in their hypervisor products. Those techniques are discussed in Section 2.2.

1.2 Cloud Gaming

In traditional client-server online games, a player sends regular updates about the state of the game to the server, and receives updates about other players in the game from the server. Nevertheless, as game logic is processed at the client instead of the server, a player may not be able to play high-end games due to the processing power of any single machine used to play games. Recently, both industry and the research community have been actively exploring the solutions to resolving this problem through cloud computing techniques.

Through utilization of elastic hardware resources[4] and widely deployed data centers, cloud computing has brought new business models for the IT industry. Specifically, it turns the idea of cloud gaming into a reality. "Cloud Gaming, in its simplest form, renders an interactive gaming application remotely in a cloud and streams scenes as a video sequence back to a thin client over the Internet" [37]. The thin client is responsible for collecting/sending a player's commands to the cloud and displaying game scenes received from the cloud. Figure 1.1 shows a deployment scenario of cloud gaming systems. A player first logs into the system via a portal server, which offers a list of available cloud games. Then the player selects a game and makes a request to play the game. Upon the receipt of a playing request, the portal server executes the selected game on an available gaming server and returns the game server's IP address to the player. Finally, the player connects to the gaming server and starts to play the selected game. With the help of virtualization, the portal server and gaming server may be able to be deployed on virtual machines, thus significantly improving the utilization of hardware resources. For instance, a high-end server can run hundreds of games concurrently with performance guarantee through virtualization.

Today, cloud gaming is showing tremendous market potential for game developers. A recent market research study[5] breaks current game market growth into three categories: boxed games, games sold online and cloud games. In particular, the market of cloud gaming is expected to expand the most: 9 times over the period of 2011 to 2016, at which time it is forecast to reach 81 billion US dollars.[6] Industry now is making the effort to develop cloud gaming systems as the market potential of cloud gaming is tremendous. The pioneers of cloud gaming, Onlive[7] and Gaikai[8] both have seen success with multi-million user bases.

[3]VMware Official Site. Website: http://www.vmware.com/. Accessed: Oct 12, 2015.
[4]Elastic hardware resources are the resources that are dynamically used to cope with dynamic workloads.
[5]Distribution and Monetization Strategies to Increase Revenues From Cloud Gaming. Website: http://www.cgconfusa.com/report/documents/cloudgaming_report_brochure.pdf. Accessed: Oct 12, 2015.
[6]Online Sales Expected to Pass Retail Software Sales in 2013. Website:http://www.neogaf.com/forum/showthread.php?t=444009. Accessed: Oct 12, 2015.
[7]Onlive Official Site. Website: https://www.onlive.com/. Accessed: Oct 12, 2015.
[8]Gaikai Official Site. Website: https://www.gaikai.com/. Accessed: Oct 12, 2015.

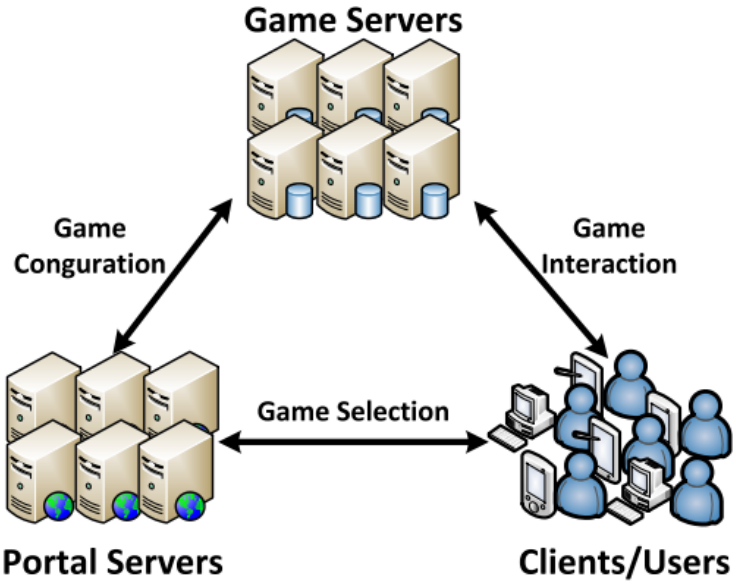


Figure 1.1: A Deployment Scenario for Cloud Gaming Systems [24]

Recently, the research community is also actively exploring the potential of cloud gaming. An open source cloud gaming system called GamingAnywhere (GA), which is developed by Huang *et al.* [23], now has become useful study material for people who want to explore cloud gaming. GA is a cross-platform which is available on Windows, Linux, OS X, and can be ported to Android and iPhone. In addition, thanks to the openness, various algorithms, standards, protocols, and system parameters can be evaluated and reviewed using GA. Moreover, GamingAnywhere is designed to be efficient, an experiment conducted on a middle-range machine with the Intel i7 Processor shows that GamingAnywhere delivers real-time 720p videos at 35 fps, which equals to 28.6 ms of processing time for each video frame, with a video quality higher than existing cloud gaming systems [23]. The architecture of GamingAnywhere is explained in Section 2.3.3.

1.3 Thesis Motivation

GPUs were originally for accelerating graphics computing, such as in gaming and video playback. Nevertheless, as modern GPUs draw more power, contain more transistors and provide at least an order of magnitude more computational performance than CPUs, GPU acceleration has extended to the basic windowing systems of operating systems and non-graphical high performance computing, such as image processing, weather forecasting and protein folding.

Recently, the demand for full GPU virtualization is increasingly raised by more and more applications. For instance, modern desktop virtualization requires GPU virtualization to support native graphical user experience in a VM. Meanwhile, cloud service vendors have started to build GPU-accelerated virtual instances for some application scenarios like cloud gaming. Only full GPU virtualization can fit the diverse requirements in those usages.

Recent hardware and software advances have realized the GPU pass-through technique, which allows a virtual machine to access a physical GPU directly and exclusively. Additionally, both industry and the research community are engaged in utilizing various strategies to implement GPU sharing, which supports multiple virtual machines to share a physical GPU with performance guarantee. VMware Inc., one of the most successful virtualization solutions vendors, has integrated GPU pass-through and two GPU sharing solutions in their hypervisor products. The performance and feasibility of these solutions in cloud gaming, however, has not been systematically characterized. Therefore, one of the primary motivations of this work is to evaluate the performance and feasibility of these solutions offered by VMware in cloud gaming scenarios, by capturing performance metrics using standard tools under various game play and virtualization scenarios. Another motivation of this work is is to perform an analysis of the hardware resource usages of these GPU virtualization solutions to find the main bottlenecks causing performance degradation.

1.4 Thesis Contributions

This thesis explores GPU pass-through and two GPU sharing virtualization solutions provided by VMware in cloud gaming scenarios, and analyses their hardware resource usages. It makes the following contributions:

- This thesis is the first systematic study to evaluate VMware's GPU virtualization solutions in cloud gaming. Firstly, the performance of each solution is measured via some graphics card benchmarks. Secondly, three cloud games are used to evaluate whether these solutions can guarantee acceptable image quality and tolerable response delay for different categories of games. Thirdly, the potential scalability of these solutions is evaluated by running two instances at the same time.
- The analysis of the hardware resource consumption of each GPU virtualization solution is also done in this thesis. Additionally, this thesis also analyzes the hardware resource consumption of the physical machine while launching each type of GPU virtualization instance.

1.5 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 describes the background and related work. Chapter 3 presents the experiments which are conducted. Chapter 4 discusses the experimental results and Chapter 5 is the conclusions.

Chapter 2

Background and Related Work

2.1 Virtualization Overview

The terminology *virtualization* was introduced in the 1960s to refer to a virtual machine in an experimental IBM M44/44X system. A layer of software is placed between the hardware layer and the operating system layer which allows the computer to support multiple different operating systems at the same time. This approach has several advantages. For instance, each virtual machine has strong isolation so that a failure in one virtual machine does not bring down any others. Additionally, check-pointing and migrating virtual machines (e.g., for load balancing across multiple servers) is much easier than migrating processes running on a normal operating system. Moreover, having fewer physical machines saves money on hardware and electricity, and takes up less rack space [41].

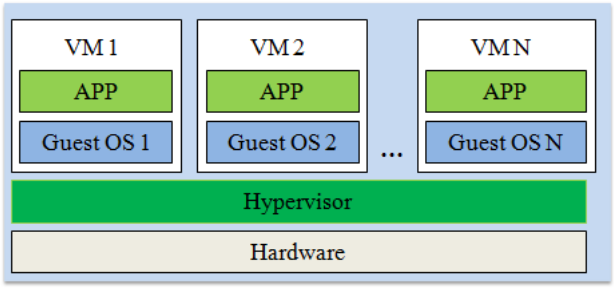
A *virtualization layer* is an essential component in virtualization as it provides the capability of using hardware resources to create multiple virtual machines with isolation and performance guarantees. As depicted in Figure 2.1, there are two types of virtualization approaches according to the implementation of the virtualization layer: the bare-metal method and the host-based approach. Figure 2.1(a) illustrates the bare-metal approach, in which a hypervisor[1] is responsible for providing hardware abstraction for the guest operating system. The hypervisor is independent of operating system, and runs on the host machine directly. Xen,[2] VMware ESXi[3] and Microsoft Hyper-V[4] are typical solutions that adopt the bare-metal approach. As shown in Figure 2.1(b), the host-based approach also utilizes the hypervisor to provide and manage virtualization. Nevertheless, the hypervisor in the host-based approach runs in a layer between host operating system and virtual machine. VMware workstation and Oracle VirtualBox are typical examples of implementations of this type.

[1] This software is also called virtual machine monitor (VMM), these two terminologies will be used interchangeably in the thesis.

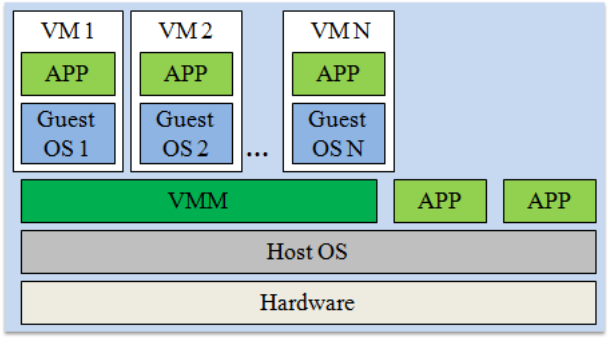
[2] Xen Introduction Page. Website: http://www.xenproject.org/. Accessed: Oct 20, 2015.

[3] VMware ESXi Introduction Page. Website: http://www.vmware.com/ca/en/products/vsphere-hypervisor. Accessed: Oct 20, 2015.

[4] Hyper-V Main Page. Website: http://www.microsoft.com/en-us/server-cloud/solutions/virtualization.aspx Accessed: Oct 20, 2015.



(a) Bare-metal hypervisor



(b) Host-based hypervisor

Figure 2.1: Bare-metal Hypervisor versus Host-based Hypervisor

2.1.1 CPU Virtualization

The x86 architecure refers to a family of backward-compatible instruction set architectures based on the Intel 8086 CPU [41]. Now it has become a standard architecture for 32-bit micro-processors, and many additions and extensions have been added to the x86 instruction set over the years, almost consistently with full backward compatibility. The x86 architecture has been widely adopted in processors from Intel, AMD and other companies.[5] Therefore, the history of x86 virtualization basically is the history of CPU virtualization. Figure 2.2 shows four privilege levels provided by the x86 architecture. Typically, user-level applications run in Ring 3, and the operating system runs in Ring 0 (the highest privilege) as it has to access the hardware resources directly. In order to virtualize the x86 architecture, a virtualization layer is required to be placed under the operating system for creating and managing virtual machine and hardware resources. In pure virtualization, the operating system is moved to Ring 1 or Ring 2 with higher privileges than user-level applications but less privilege than the hypervisor in Ring 0. However, without special hardware support it is not possible to trap all privileged and sensitive x86 instructions[6] issued by the operating system at runtime, which makes pure x86 virtualization using the classical trap-and-emulate technique impossible

[5]Introduction to x86 architecture. Website: http://en.wikipedia.org/wiki/X86. Accessed: Oct 20, 2015.

[6]Privileged instructions are instructions that can only be executed in the highest level (Ring 0), and sensitive instructions are instructions that behave differently when executed in different levels.

without hardware modifications [4]. For instance, certain instructions issued by the guest operating system outside Ring 0 can return a value which indicates the current privilege level. Therefore, the guest operating system can determine that it is not running in Ring 0 in this way, causing a problem known as ring aliasing. With the development of hardware-assisted virtualization implementations such as Intel VT-d [1] and AMD-V, however, these issues now have been resolved [32]. The basic idea of these techniques is to create a container in which virtual machines can be deployed and executed. When the guest operating system is started up in a container, it continues to run there until it causes an exception and traps to the hypervisor, for example, by executing an I/O instruction. The set of operations that cause a trap is controlled by a hardware bitmap set by the hypervisor. With these extensions, the classical trap-and-emulate virtual machine approach becomes possible [41]. Hardware-assisted virtualization is not the only possible approach, and currently there exist three main CPU virtualization techniques for the x86 architecture:

- Full Virtualization Using Binary Translation,
- Para-virtualization, and
- Hardware-assisted Virtualization.

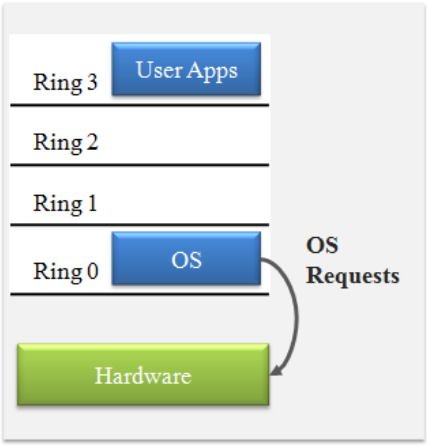
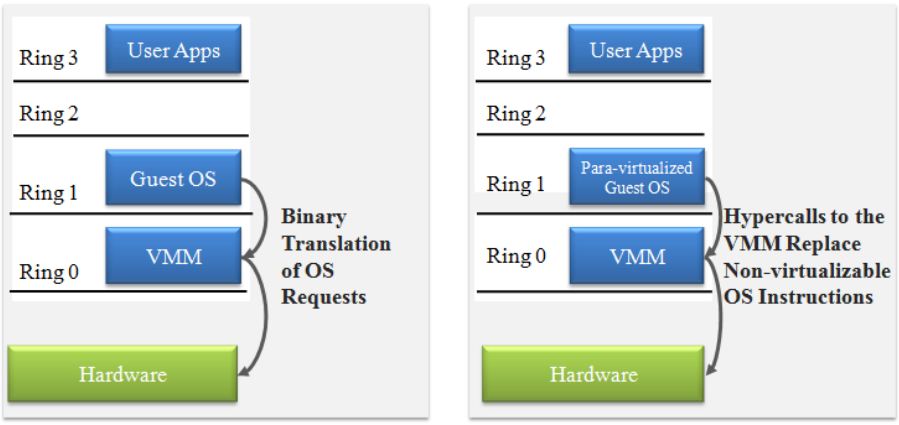
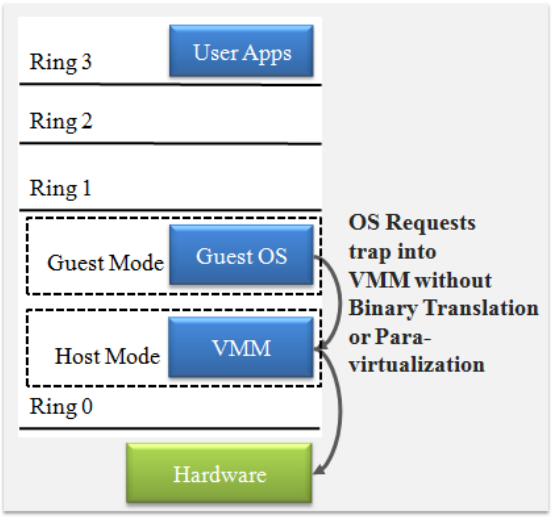


Figure 2.2: Privilege Levels of x86 Architecture Without Virtualization

Full virtualization [32] virtualizes the x86 operating system using a combination of direct execution technique and binary translation. As shown in Figure 2.3(a), with the binary translation technique the VMM is run in Ring 0 and the guest OS in Ring 1. Using binary translation, all privileged instructions issued by the guest OS now are translated by the hypervisor and non-virtualizable instructions are replaced with new sequences of instructions (emulated version of these instructions). Full virtualization makes the guest OS fully separated from the underlying hardware through a virtualization layer. The guest operating system has no idea it is being virtualized, which means no modification is required. Figure 2.4(a) shows the simplified architecture of full virtualization. Although there are some performance overheads caused by binary translation, full virtualization provides good isolation and security for virtual machines, as well as simplified live migration of virtual machines.



(a) Privilege-level View of Full Virtualization (b) Privilege-level View of Para-virtualization



(c) Privilege-level View of Hardware-assisted Virtualization

Figure 2.3: Full Virtualization versus Para-virtualization versus Hardware-assisted Virtualization

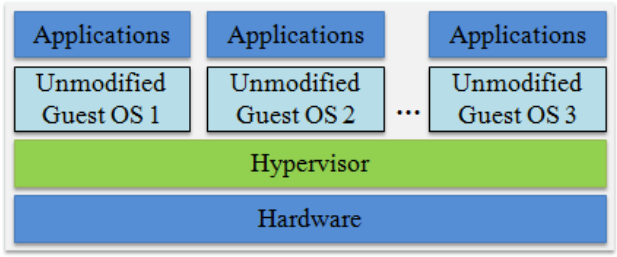
Para-virtualization [32] is another approach to virtualizing the x86 CPU. The guest OS in para-virtualization is aware of the occurrence of virtualization, which makes the OS modification necessary. As shown in Figure 2.3(b), the VMM is runs in Ring 0, while the guest OS runs in Ring 1.[7] In para-virtualization, non-virtualizable instructions in the guest OS now are replaced with hypercalls that are used to communicate with the hypervisor. In one approach to para-virtualization shown in Figure 2.4(b), a unique host OS running in a special domain (Domain 0)[8] is responsible for managing hardware drivers and interacting with the hypervisor. Domain 0 is parallel with other domains used to run the guest OSs. Each guest OS accesses hardware resources by issuing hypercalls to the hypervisor. Then the hypervisor handles these hyper-calls by calling

[7]In para-virtualization, the guest OS runs either in Ring 1 or Ring 2.

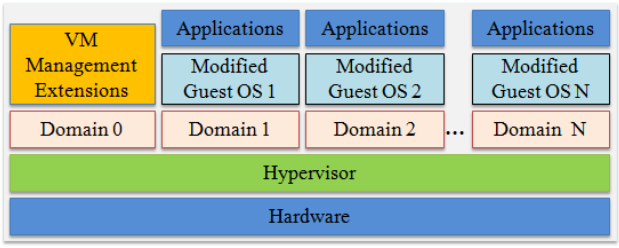
[8]A domains in para-virtualization is a running instance of a virtual machine, while Domain 0 is a special virtual machine that runs a para-virtualized operating system which is used to manage virtualization.

corresponding hardware drivers in Domain 0. Finally, all requests are processed by corresponding hardware drivers. Para-virtualization ensures lower virtualization overheads than full virtualization. Its compatibility and maintainability, however, is poor as it only supports modified operating systems.

Hardware-assisted virtualization makes use of those hardware features provided by hardware vendors for simplifying CPU virtualization. In 2006, Intel and AMD provided Intel Virtualization Technology (VT-d) and AMD-V respectively, to simplify virtualization. Take AMD-V as example, the VMM and the guest OS in hardware-assisted virtualization run on the same privilege level but in different modes as illustrated in Figure 2.3(c). The VMM runs in the host mode that refers to previously architected x86 execution environment, while the guest OS runs in a new, less privileged execution mode called guest mode. Additionally, a new instruction, vmrun, is introduced to transfer from the host mode to the guest mode. Although the first generation of hardware-assisted virtualization fails to offer performance advantages over full virtualization and para-virtualization, it is considered a promising technique that may be able to significantly improve CPU virtualization performance one day.



(a) Full Virtualization



(b) Para-virtualization

Figure 2.4: Full Virtualization versus Para-virtualization [32]

2.1.2 Memory Virtualization

Memory is another critical hardware resource to be virtualized. Memory virtualization refers to dynamically allocating physical memory to virtual machines and managing the mapping between virtual pages in virtual machines and physical pages. Modern x86 CPUs use a memory management unit (MMU) and a translation look-aside buffer (TLB) to implement virtual memory address translation. In virtual memory systems, an

application is assigned a virtual address space that is organized as a set of virtual pages, while the mappings between these virtual pages and physical pages are under the control of the operating system. Memory virtualization for virtual machines is similar to the virtual memory sub-system in modern operating systems. As shown in Figure 2.5, a guest OS provides a virtual memory sub-system for the processes running on top of it, and the MMU must be virtualized so that the guest OS's physical memory can be mapped to actual machine memory. To avoid two levels of mapping on every access, the VMM utilizes TLB hardware to map virtual memory directly to machine memory. In addition, shadow page tables are used on TLB misses. A shadow page table is maintained for each process of each VM. These shadow page tables control which pages of machine memory are assigned to each process of each VM. When the guest OS updates one of its page tables, the VMM updates its corresponding shadow page table. Although MMU virtualization introduces some virtualization overhead, it is an area that the second generation hardware-assisted virtualization focuses on.

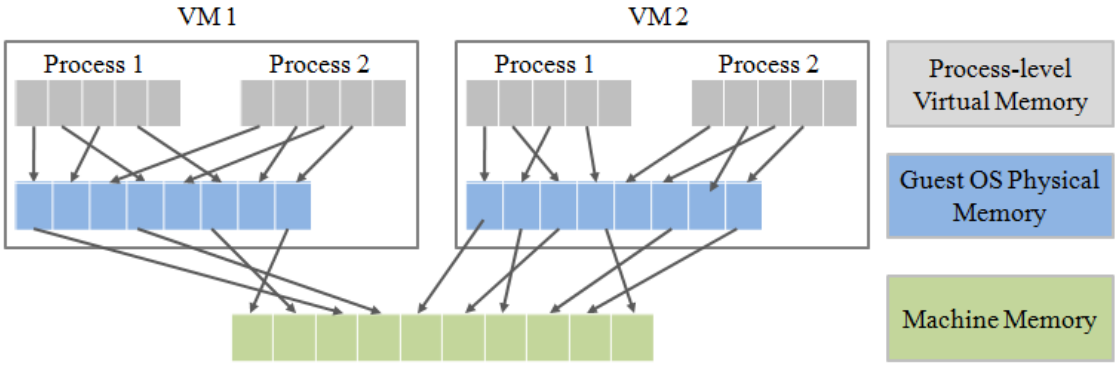


Figure 2.5: Memory Virtualization

2.1.3 I/O Virtualization

I/O virtualization involves managing the mapping of I/O requests between virtual devices and the shared physical hardware. There are four approaches to virtualize I/O devices:

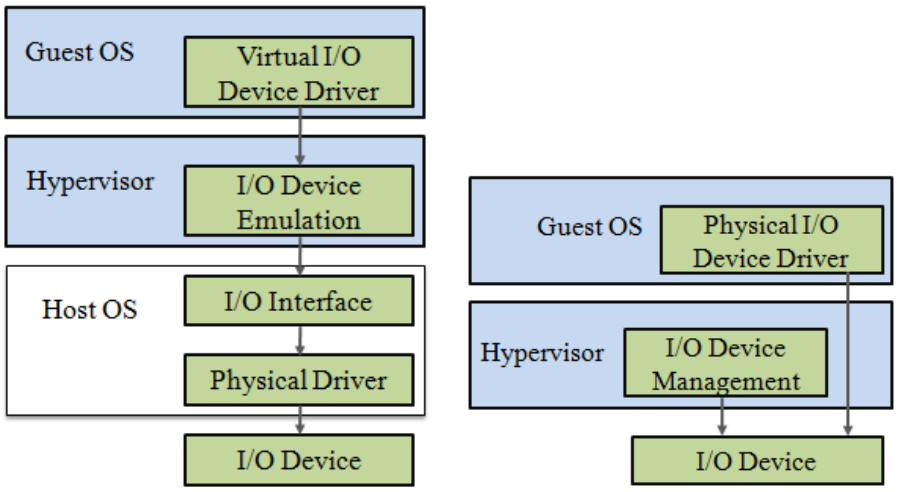
- Full Virtualization,
- Para-Virtualization,
- Device Emulation, and
- I/O Pass-through.

Full virtualization for I/O virtualization is similar to that for CPU virtualization. The hypervisor multiplexes physical I/O devices, and provides virtual device interfaces for virtual machines. All I/O instructions issued by the guest OS are trapped and handled by the hypervisor, which drives corresponding devices to complete these I/O requests. I/O para-virtualization also follows the basic principles in CPU para-virtualization. In the context of the approach to para-virtualization discussed previously for CPU virtualization, a back-end

driver (a physical driver) is installed in Domain 0 to access the physical device, and a front-end driver is installed in the guest OS, which handles and passes the guest OS's I/O requests to the back-end driver through the hypervisor. Upon receipt of the guest OS's I/O requests, the back-end driver accesses corresponding devices to handle the requests and sends the results back to the corresponding virtual machine.

Device emulation is a general solution which has been widely used to virtualize different devices. It is normally adopted in host-based hypervisors. As shown in Figure 2.6(a), I/O requests issued by the guest OS are intercepted by the hypervisor and passed to a process in the host OS, which invokes system calls to handle received I/O requests. The main overhead in such an approach is the context switches which occur between the guest OS and the VMM, kernel space and the user space, and between the emulation application and the host OS kernel. Therefore, the main optimization in this method is to reduce context switches.

Figure 2.6(b) shows a simplified framework of direct I/O pass-through. Unlike the three approaches described above, virtual machines in direct I/O virtualization can access hardware devices directly without going through the hypervisor. The idea of I/O pass-through is introduced by Liu *et al.* [31]. The device is still controlled by a physical driver in Domain 0, but the guest OS is allowed to access it directly. Although this approach improves performance and enables the guest OS to take full advantages of all hardware functionality, its use requires addressing two main issues. First, the hypervisor should ensure isolation so that a virtual machine cannot access memory allocated to other VMs. Second, the I/O device being virtualized must provide the ability and interfaces to allow multiple guest OSs to access it concurrently.



(a) I/O Device Emulation (b) Direct I/O Pass-through

Figure 2.6: I/O Device Emulation and Direct I/O Pass-through

2.1.4 GPU Virtualization

Although most computer resources have been well virtualized, difficulties remain in full virtualization of the GPU. GPUs are bad at handling multiple GPU-intensive applications concurrently, even when run on a bare-

metal system. Traditionally, multi-tasking on CPUs have been well supported with reasonable preemption latency and through overhead through context switching. Nevertheless, the same multi-tasking strategy on GPUs incurs higher overheads compared to CPUs, due to the large context in GPUs [34]. During the context switch, a GPU has to save and restore up to 256 KB of resgiter file and 48 KB of on-chip scratch-pad memory per GPU core, which can take up to 44 *us* in the latest NVIDIA GPU architecture, Kepler GK 110[9], assuming the peak memory. It is a high overhead compared to the context switch time of less than 1 *us* on modern CPUs. Not only the OS kernel has to suffer from a long preemption latency, the GPUs also waste execution resources during the process of context switching [40]. Additionally, GPU memory is used in gaming and graphics applications to store textures and shadow maps, as well as the frame buffer and the depth buffer for rendering a frame. This is completely different for each gaming application, so no resources can be shared.

Recent hardware advances have allowed virtualization systems to implement one-to-one mapping between a virtual machine and a physical GPU. In addition, GPU sharing (which can create multiple virtual GPUs) is emerging as a popular research area. Current GPU virtualization solutions can be roughly separated into two major classes, namely, front-end virtualization and back-end virtualization. The former is based on the device emulation technique [3] or API remoting [39], while the latter relies on PCI pass-through [13].

Front-end Virtualization

Front-end virtualization requires no GPU vendor- or model-specific requirements as it realizes GPU virtualization at a relatively high level and runs the graphics drivers on the host. In Front-end virtualizaiton, the guest OS does not have the direct access to physical GPUs. Instead, the VM's accesses to physical GPUs are totally mediated through provided drivers on the host OS. Multiplexing is easily implemented with such a technique because it is based on software and multiple "contexts" for applications are allowed by current GPUs. However, since the virtualization overhead of such a solution is relatively high compared to the bare-metal system, it might not be suitable to apply such a technique in some graphics-sensitive application scenarios, such as virtual desktop and cloud gaming.

Front-end virtualization solutions typically are based on device emulation or API remoting. Device emulation technology emulates the GPU and the virtual GPU synthesizes host graphics operations by guest device drivers, while API remoting employs a front-end driver to forward the high level API calls from VMs to the host OS. Device emulation fails to fit today's requirements as it is very complex and has extremely low performance, while API remoting faces the challenge of supporting full features due to the complexity of the modification of the guest graphics software stack. Moreover, the performance of API remoting largely depends on the applications and how well API remoting is implemented. Bandwidth-intensive and latency-sensitive applications may suffer from more serious performance degradation than the computation-intensive applications. When regarded as computing resources, however, a modern GPU equipped with thousands

[9]NVIDIA Kepler GK 110: Nextgeneration CUDA Compute Architecture Website: https://www.nvidia.com/content/PDF/kepler/NV_DS_Tesla_KCompute_Arch_May_2012_LR.pdf. Accessed: March 21, 2016.

of processing elements is more effective than general-purpose CPUs in many application scenarios like high performance computing (HPC), image processing and weather forecasting. Therefore, API remoting is able to play a key role in such scenarios as it brings significant GPU acceleration in these areas.

There have been some approaches that realize the virtualization of CUDA[10] runtime APIs for VMs including rCUDA [15], vCUDA [38], gVirtuS [18] and GViM [19]. These solutions typically comprise two components: front-end middleware and back-end middleware. The front-end middleware is installed on the VM and the back-end middleware with direct access to the hardware is installed inside the hypervisor. Remote API calls are forwarded from the front-end middleware to the back-end middleware. Upon the receipt of remote API calls, the back-end middleware drives physical GPUs to process them and sends results back.

As shown in Figure 2.7(a), the rCUDA framework can be split into two software modules: client middleware and server middleware. The client middleware consists of a collection of wrappers that forward CUDA API calls to the server middleware and retrieve results back. The server middleware acts as a service and runs on a computer equipped with a physical GPU. It receives remote API calls, executes them locally, and then sends results back to the client middleware. In addition, TCP sockets are utilized to let the client middleware and the server middleware communicate with each other [15].

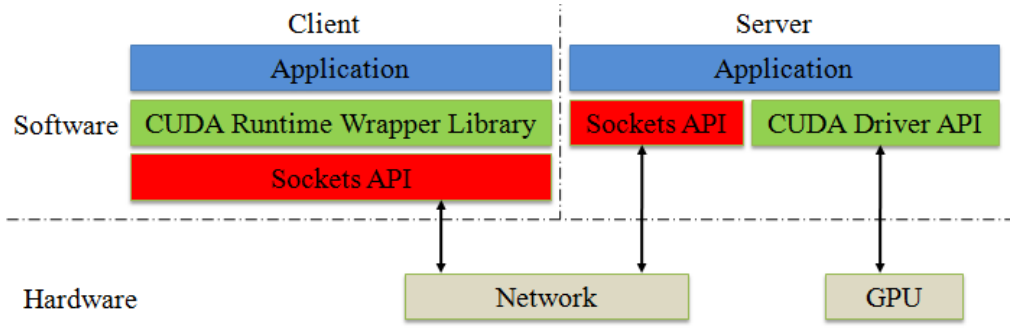
Like the rCUDA framework, vCUDA also adopts a client/server architecture. As depicted in Figure 2.7(b), vCUDA consists of three components: vCUDA library, virtual GPU and vCUDA stub in the host OS. The vCUDA library is in charge of intercepting and redirecting API calls from applications to a vCUDA stub. The virtual GPU is represented as a database maintained by the vCUDA library, and it provides GPU contexts and a complete view of underlying hardware for CUDA applications. The vCUDA stub is responsible for receiving and accomplishing remote API calls, and returning results back [38].

Figure 2.7(c) shows the architecture of gVirtuS, in which the front-end module (a wrapper CUDA library) residing in the guest OS intercepts CUDA calls issued by applications and forwards them to the back-end module. The back-end unpacks library functions, maps memory pointers, executes functions on the host's GPU, and returns results to the front-end module. The architecture of gVirtuS is similar to rCUDA's except for its communicator. It is designed to work with a pluggable communication component independent of the hypervisor. Therefore, an efficient communicator can reduce the overheads of remote execution of CUDA calls.

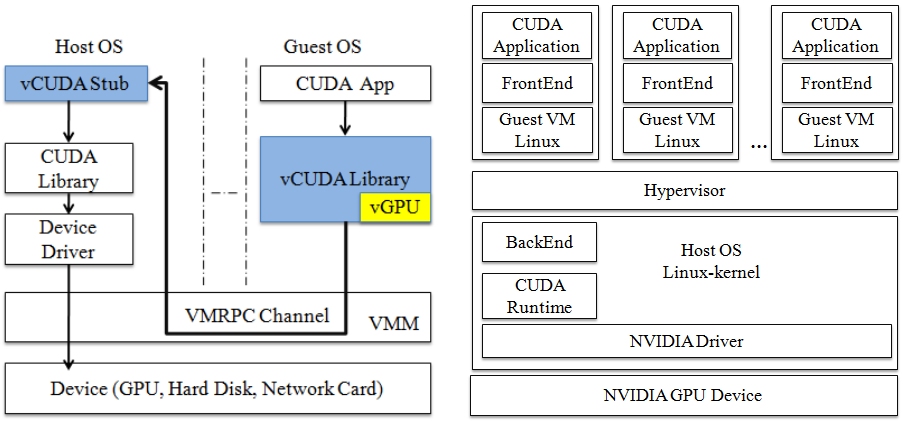
GViM is a system designed for managing resources of a general purpose system accelerated by graphics processors. It uses Xen-specific mechanisms for the communication between the front-end and the back-end middleware. A GPGPU platform virtualized by GViM enables consolidation of graphics processors. The evaluation [19] with a Xen-based implementation of GViM shows efficiency and flexibility in system usage with only small performance degradation for virtualized vs. non-virtualized solutions.

In addition to improving GPU acceleration in general purpose computing, some front-end solutions also make an attempt to realize GPU virtualization for leveraging hardware rendering acceleration. Lagar-Cavilla

[10]CUDA is a programming model proposed by Nvidia that exposes the Nvidia GPU hardware for GPGPU computing.



(a) The Framework of rCUDA [15]



(b) The Framework of vCUDA [38]

(c) The Framework of gVirtuS [18]

Figure 2.7: rCUDA versus vCUDA versus gVirtuS

et al. [28] proposed VMGL, a cross-platform OpenGL virtualization solution which is both VMM and GPU independent. Unlike CUDA-enabled virtualization solutions which aim to make remote CUDA API calls realistic, VMGL pursues the virtualization of the OpenGL library. It enables multiple applications in multiple VMs to utilize hardware rendering acceleration, alleviating the problem of limited virtualization capability of a growing class of graphics-intensive applications. Additionally, it also offers suspending and resuming capabilities for applications. Figure 2.8 shows the architecture of VMGL, in which the VMGL library, VMGL stub and VMGL X server extension are three main components. The VMGL library inside the guest OS works as a replacement for standard or vendor-specific OpenGL implementations. When an OpenGL application inside a VM starts, the VMGL library creates a VMGL stub on the host to perform as a sink for OpenGL commands. The VMGL stub acts on the behalf of the virtualized application to obtain direct rendering capabilities. Once an application issues an OpenGL command, the VMGL library forwards the command to the VMGL stub via a network transport. In this case, each application owns a unique VMGL stub, and each stub runs as a separate process, providing graphics rendering capability for multiple applications concurrently. The Blink system proposed by Hansen [20] is a similar system to VMGL, which multiplexes sophisticated graphical contents from multiple virtual machines onto a shared GPU. The Blink

display server is a user-level application that runs inside a Linux guest VM. It accesses graphics hardware using commercially developed device drivers. The client inside the guest VM accesses the physical GPU by communicating with this server. Moreover, The Blink extends the display list abstraction[11] into more general and flexible BlinkGL stored procedures. These stored procedures are able to handle simple user interactions like redrawing the mouse cursor or highlighting a pushbutton in response to a mouse rollover. Moreover, Blink server contains a just-in-time (JIT) compiler that can convert BlinkGL into machine code, reducing the translation costs.

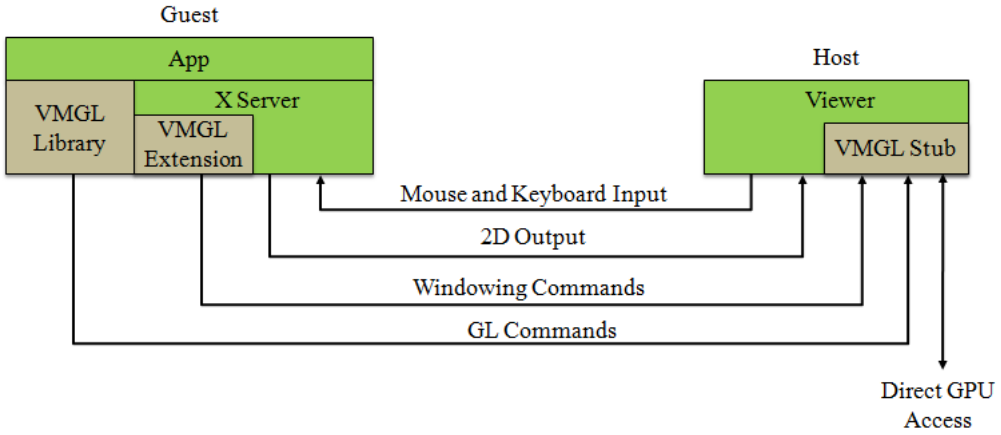


Figure 2.8: The Architecture of VMGL [28]

Back-end Virtualization

The back-end virtualization technique is another solution that can achieve higher GPU virtualization performance than front-end technology [14]. Back-end technology executes a graphics driver inside a virtual machine with virtualization boundary between the driver stack and a physical GPU. Through the back-end method, a virtual machine can directly interact with a physical GPU. This technique gains higher performance than front-end virtualization, as direct access to native physical GPUs is excellent for achieving good fidelity: a VM can utilize full features of GPU abilities. Its multiplexing, however, is a serious challenge. As modern GPUs normally are bad at multi-tasking graphics-intensive applications, there remains difficulties in realizing multiplexing with the back-end technique to enable multiple VMs to share a physical GPU concurrently with isolation and performance guarantees.

One back-end virtualization technique that has been adopted in industry is fixed pass-through, which dedicates a physical GPU to a single VM and offers full features and the best performance [42]. Recent chipset features, like Intel's VT-d and AMD-Vi, make it realistic to realize fixed pass-through without requiring any special requirement of the GPU's programming interfaces. Recently, many virtualization solution vendors have proposed their fixed pass-through techniques in their hypervisor products. For instance, VMware Inc.[12]

[11] Display list abstractions are macro sequences of OpenGL commands.
[12] VMware Official Site. Website: http://www.vmware.com/. Accessed: Oct 20, 2015.

developed a technique called vDGA to enable a physical GPU to be mapped to a single VM. XenServer has also incorporated a similar technique into its hypervisor product to support fixed GPU pass-through. Although it provides the best performance, fixed pass-through is an expensive solution as it completely gives up multiplexing.

Another back-end solution is mediated pass-through, which just dedicates a context to a virtual machine rather than an entire GPU. This solution offers full features and multiplexing capability with good performance. But there are still two challenges that need to be resolved. First, GPU hardware must be designed and implemented in a way that can manage multiple contexts, and so that the contexts can be mapped into different virtual machines with low overheads. Second, the host/hypervisor must be able to allocate and manage multiple GPU contexts using graphics drivers. Currently, mediated pass-through is rarely adopted as difficulties remain. From an extensive literature search, it appears that gVirt developed by Tian *et al.* [42] is the first commercial GPU virtualization implementation with 1) full GPU virtualization which runs a native graphics driver in the guest OS, and 2) mediated pass-through that ensures good GPU performance and fidelity for each VM. Figure 2.9 illustrates the overall gVirt architecture based on Xen hypervisor. Each VM runs a native graphics driver to access the physical GPU resources directly. gVirt Mediator in Domain 0 is responsible for allocating and managing virtual GPUs for VMs, and using hypercalls to access physical GPU. This mediator also manages a GPU scheduler, which is parallel with the CPU scheduler in Xen, to schedule the execution of virtual GPUs. The gVirt stub module is in charge of trapping and forwarding guest access of certain GPU resources. All the trapped GPU accesses are forwarded to the mediator. Then it invokes hypercalls to access physical GPU resources.

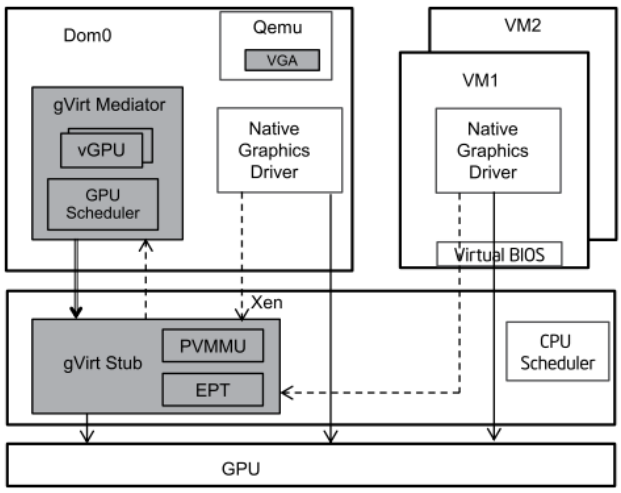


Figure 2.9: The Architecture of gVirt [42]

Performance Studies of GPU Virtualization

Recently the potential and performance of existing GPU virtualization solutions in high performance computing have been studied in the literature. Duato *et al.* [16] modeled rCUDA over a series of high throughput networks for assessing the influence of the underlying network on the performance of rCUDA in high performance clusters, by analyzing the traces of two different case studies over two different networks (1 Gbps Ethernet and 40 Gbps InfiniBand networks). Through the study, they found rCUDA performed almost as efficiently as a local GPU when using high performance interconnects like 40 Gbps Infiniband networks.

Vinaya *et al.* [43] presented a detailed evaluation of GPU acceleration in rCUDA, gVirtuS and Xen. They utilized a subset of CUDA SDK examples which provide a comprehensive coverage of CUDA APIs that compromise simple and complex benchmarks, to compare the three frameworks in terms of their performance, and characteristics of fidelity[13], interposition[14] and multiplexing. Their experimental results showed that although Xen with GPU pass-through enabled lost the ability to multiplex, it provided maximum fidelity and performance compared to the other two solutions. Moreover, rCUDA shows greater fidelity but lower performance than gVirtuS. Furthermore, none of these solutions provide desirable interposition features.

Walters *et al.* [44] characterized the performance of the GPU pass-through mode provided by VMware ESXi, KVM, Xen and Linux Containers for CUDA and OpenCL applications. Through the utilization of a series of micro-benchmarks as well as scientific and big data applications, they demonstrated that GPU pass-through achieved near-native performance in high performance computing across four major hypervisors.

The performance study conducted by Shea *et al.* [36] evaluated the performance of real world gaming and ray-tracing applications in a VM with GPU pass-through for both Xen and KVM. They found that although the VMs were accelerated with dedicated physical GPUs, gaming applications performed poorly when virtualized as compared to the non-virtualized bare-metal baseline. Moreover, their detailed performance analysis on KVM revealed that a memory bottleneck was the main cause for the performance degradation.

2.2 VMware's GPU Virtualization Solutions

Rich applications are presenting rising demands for full GPU virtualization with good performance, full features, and sharing capability. Generally, the use cases in these rich applications can be divided into three categories as the following [26]:

- **Knowledge Workers:** Knowledge Workers include office workers and executives, typically using less graphics-intensive applications such as Microsoft Office, web browser and other non-specialized end-user experience applications. The key areas of importance for this type of user are office productivity applica-

[13] In scientific modeling and simulation, fidelity is the degree to which a model or simulation reproduces the behaviour and state of a real world object, feature or condition. Here the fidelity denotes how closely a virtual machine is implemented to a real machine.

[14] Interposition concerns the abstraction which is used to deliver secure isolation, resource management, virtual machine portability and many other features by separating the guest from physical hardware dependencies.

tions, rich web experience, and fluid video playback. They expect a similar smooth and fluid experience that can be achieved natively on today's graphic accelerated devices such as desktop PCs.

- **Power Users:** Power Users are those users who need to run more graphics-intensive applications, including image editing software such as Adobe Photoshop,[15] and mainstream computer-aided design (CAD) applications such as Autodesk AutoCAD.[16] Those applications require more GPU resources with full support for graphics APIs such as OpenGL and DirectX.
- **Designers:** Designers are those users with the need to run highly graphic-intensive applications such as high-end CAD modeling software, for example Autodesk Inventor.[17] Traditionally designers utilize desktop workstations and it is difficult to incorporate applications in this category into virtual deployments due to the need for high-end graphics and certification requirement of those applications.

As one of the most successful virtualization solution vendors in the world, VMware Inc. has been making great effort to improve GPU virtualization and has provided four solutions for satisfying various demands of GPU virtualization in its product. The default GPU virtualization solution in VMware ESXi is called Soft 3D, which is a software-based virtualization method implemented via device emulation. VMware also provides a light-weight GPU-sharing solution called Virtual Shared Graphics Acceleration (vSGA) using front-end virtualization technology, and a fixed GPU pass-through solution called Virtual Dedicated Graphics Acceleration (vDGA). Moreover, it recently realized dedicated GPU pass-through by introducing Nvidia's vGPU technology into its products. All these three solutions are available in VMware ESXi and all GPU features brought by these solutions can be utilized through VMware Horizon View,[18] a virtual desktop infrastructure that provides end users access to virtual desktops and applications created by VMware virtualization platforms.

Soft 3D is the default GPU virtualization solution that VMware offers. It is a software-based method that does not requires any physical GPUs to be installed on the VMware ESXi host. This solution only requires an emulated graphics driver automatically installed on virtual machines. It provides support for DirectX 9c and OpenGL 2.1 and supports both software 2D and 3D rendering in Windows 7 virtual desktops. Nevertheless, Soft 3D is not suitable for running applications that need 3D features as the performance of device emulation is not good while running this type of application. Therefore, it is only suitable for running applications that fall into the use case of Knowledge Workers.

vSGA is the first GPU sharing solution introduced with VMware Horizon View 5.2. It is differentiated from Soft 3D in that it is a hardware-based solution that provides hardware-accelerated 3D graphics by enabling multiple virtual machines to share a physical GPU installed in the ESXi host. As shown in Figure 2.10, a physical GPU and its driver need to be installed in the VMware ESXi host. Each virtual machine installs

[15]Adobe Photoshop Family. Website: http://www.adobe.com/products/photoshopfamily.html. Accessed: Dec 02, 2015.
[16]Autodesk AutoCAD Official Website: http://www.autodesk.com/products/autocad/overview. Accessed: Dec 02, 2015.
[17]Autodesk Inventor Official Website. Website: http://www.autodesk.com/products/inventor/features/all. Accessed: Dec 02, 2015.
[18]This software is also known as VMware View, these two terminologies will be used interchangeably in the thesis.

and utilizes a proprietary VMware vSGA 3D driver that communicates with the physical graphics driver in the VMware ESXi host. VMware vSGA can improve performance for the use case of Knowledge Workers by providing high levels of consolidation of users across a physical GPU. Nevertheless, it mostly limited to this use case since it does not provide a wide range of graphics API support.

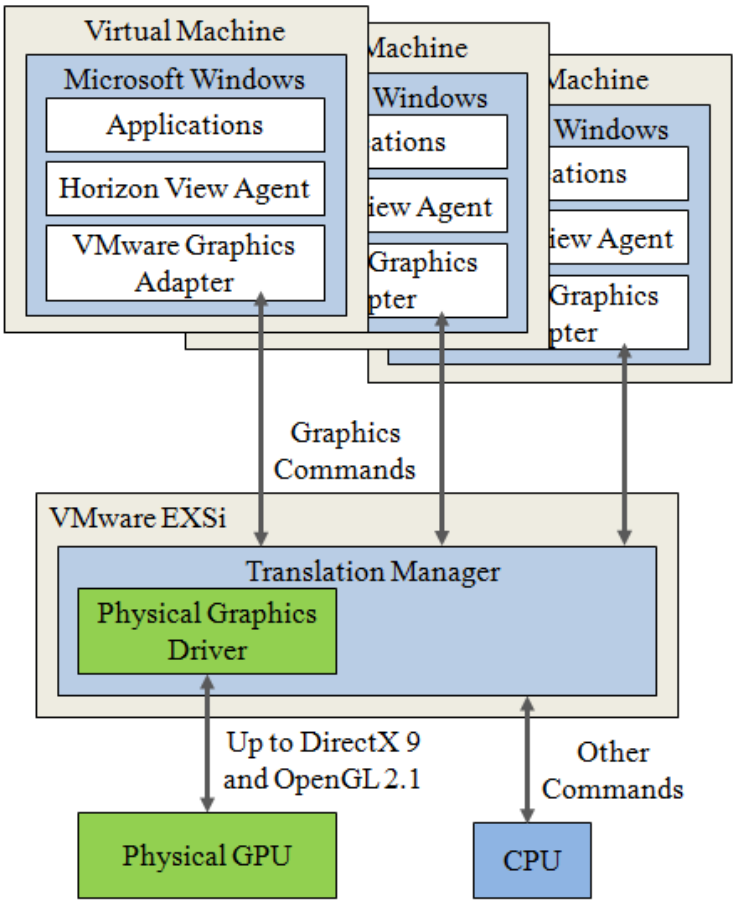


Figure 2.10: VMware vSGA [26]

vDGA is a fixed GPU pass-through solution introduced with VMware Horizon View 5.3. It is a graphics-acceleration capability that delivers high-end workstation graphics for use cases where a dedicated GPU is required. This method dedicates a single GPU to a single virtual machine for high performance. An example of using VMware vDGA is illustrated in Figure 2.11. In this case, an Nvidia GRID K2 equipped with two high-end Kepler GPUs is installed in the ESXi host. No physical graphics driver is needed in the ESXi host. To enable graphics acceleration, an appropriate physical Nvidia GRID driver needs to be installed on a virtual machine. With this configuration, the virtual machine can have direct and exclusive access to the physical GPU. Although this technology offers a user fully dedicated access to a single GPU, it sacrifices the consolidation as the physical GPU occupied by the vDGA VM cannot be accessed by other VMs. For instance, in this case, since the Nvidia GRID K2 has two physical GPUs, only two virtual machines can utilize vDGA technology. Overall, this technology meets the needs in all use cases and offers the highest

level of performance for users with the most intensive graphics computing needs. Moreover, the wide range of API support such as OpenGL 4.4, Microsoft DirectX 9, 10, or 11, and Nvidia CUDA 5.0, makes it able to run high-end 3D applications.

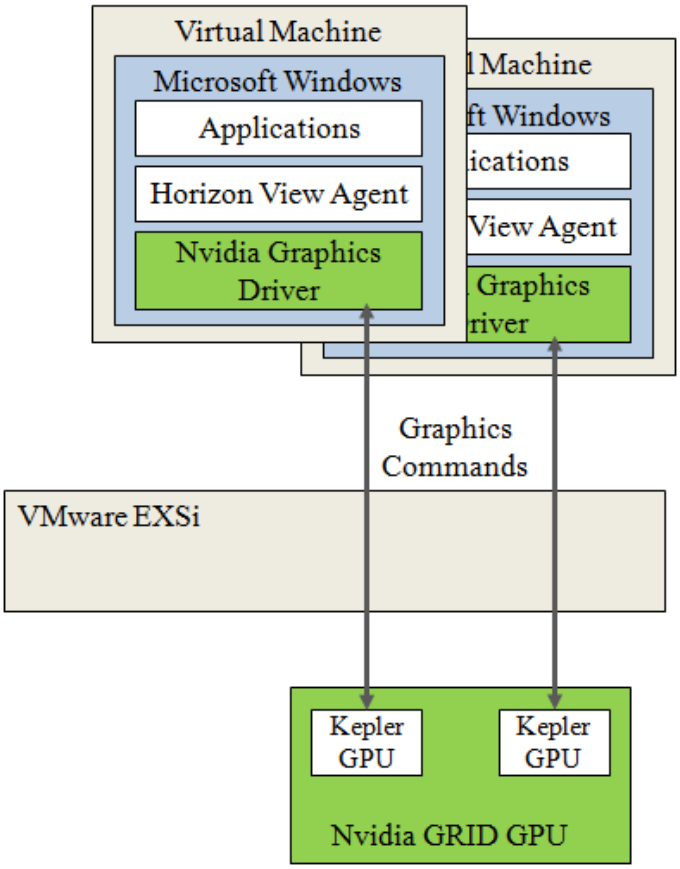


Figure 2.11: Vmware vDGA [26]

Another GPU sharing solution, vGPU technology, is supported and introduced in VMware Horizon View 6.0 and VMware ESXi 6.0. Like vDGA, vGPU brings the benefit of wide API support and native graphics performance but greater scalability. vGPU is essentially a dedicated GPU pass-through solution based on hardware acceleration. Figure 2.12 shows an instance that utilizes vGPU to realize GPU sharing. An Nvidia GRID K2 is installed in the ESXi host, and a vGPU manager is also required to be installed in the ESXi host to manage the states of each virtual GPU. Like vDGA, an appropriate Nvidia GRID graphics driver needs to be installed in the virtual machine, so that all graphics commands can be passed directly to the physical GPUs without any translation help from the ESXi host. Figure 2.13 illustrates the internal architecture of Nvidia GRID vGPU, from which we can see that each vGPU obtains its own frame buffer allocated out of the physical GPU frame buffer at the time it is created. The vGPU frame buffer is managed by the Nvidia vGPU manager installed in the ESXi host. Each vGPU retains exclusive use of its vGPU frame buffer until it is destroyed. Additionally, all vGPUs within the same physical GPU share access to the GPU engines,

including the graphics (3D), and video decode and encode engines.

vGPU technology has better performance than vSGA and higher consolidation ratios than vDGA. Additionally, it fits the demands in the use cases of Knowledge Workers, Power Users and even Designers. One drawback of this technology, however, is that high-end applications might need to obtain certification from Nvidia Inc. before running on any vGPU instances.

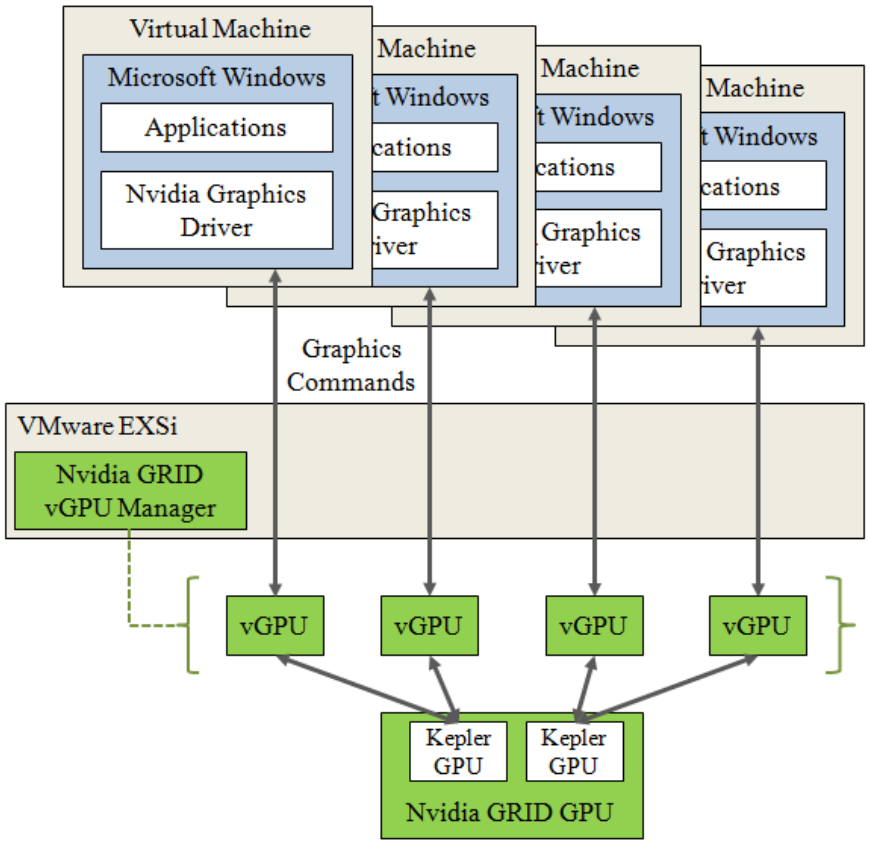


Figure 2.12: VMware Horizon View with Nvidia GRID vGPU [26]

vGPU technology is only available on specific Nvidia graphics cards, namely, Nvidia GRID K1 and GRID K2. Table 2.1 shows the main specifications for GRID K1 and K2. K1 consists of four entry-level Kepler GPUs, each of which is equipped with 192 CUDA cores and 4 GB DDR3 as video memory. K2 has two high-end Kepler GPUs, and each of them is configured with 1526 CUDA cores and 4 GB DDR5 as video memory. Additionally, K1 and K2 have a wide range of API support. Both of them support the latest version of OpenGL, Microsoft DirectX and Nvidia CUDA. K1 and K2 can support several vGPU profiles as shown in Table 2.2. Each vGPU profile has a fixed amount of video memory, number of supported displays per user, and maximum resolution per user. In addition, the vGPU profiles exhibit great flexibility as they are targeted at different classes of use case.

Table 2.1: Main Specifications for Nvidia GRID K1 and K2 [29]

Product Name	GRID K1	GRID K2
Number of GPUs	4 GK107 GPUs	2 GK104 GPUs
CUDA Cores	768 (192 / GPU)	3072 (1536 / GPU)
Memory Size	16 GB DDR3 (4GB / GPU)	8 GB DDR5 (4GB / GPU)
OpenGL	4.x	4.x
Microsoft DirectX	Up to 11	Up to 11

Table 2.2: vGPU Profiles [29]

Graphics Board	**Virtual GPU Profile**	**Graphics Memory (MB)**	**Maximum Displays Per User**	**Maximum Resolution Per User**	**Maximum Users Per Graphics Board**	**Use Case**
Nvidia GRID K1	K180Q	4,096	4	2560x1600	4	Entry Designer
	K160Q	2,048	4	2560x1600	8	Power User
	K140Q	1,024	2	2560x1600	16	Power User
	K120Q	512	2	2560x1600	32	Power User
Nvidia GRID K2	K280Q	4,096	4	2560x1600	2	Advanced Designer or Engineer
	K260Q	2,048	4	2560x1600	4	Designer, Engineer or Power User
	K240Q	1,024	2	2560x1600	8	Designer, Engineer or Power User
	K220Q	512	2	2560x1600	16	Designer or Power User

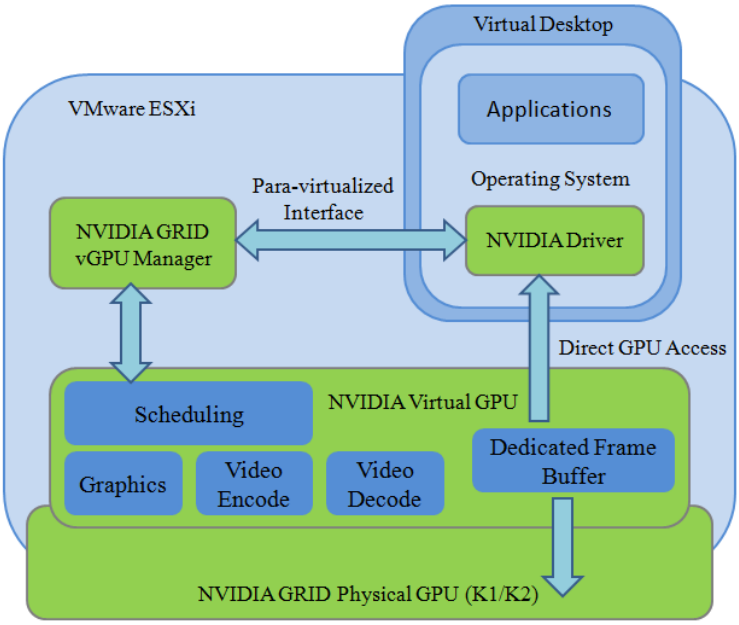


Figure 2.13: Nvidia GRID vGPU Internal Architecture [29]

2.3 Cloud Gaming

2.3.1 Overview of Cloud Gaming

Cloud computing has drastically changed existing operations and business models of IT industry because of its unparalleled scalability and reduced costs of capital and equipment maintenance. Existing applications, from document sharing to media streaming, have experienced a great benefit from cloud computing platforms, in terms of system efficiency and usability. Recently, the advances of cloud computing technology have made it realistic to offload complex tasks like graphics-intensive 3D rendering to the cloud, which turns the idea of cloud gaming into a reality and significantly facilitates its development.

As shown in Figure 2.14, a cloud gaming platform can be split into several modules. The thin client interaction module is used to receive the players' commands. The commands are converted into appropriate in-gaming actions by the game logic. Graphics rendering is responsible for processing game world changes into rendered scenes. The rendered scenes are encoded and compressed as a video stream by the video encoder module and streamed back to the thin client via a real-time streaming module. Finally, the video stream is decoded and rendered by the video decoder in the thin client.

Compared to traditional gaming platforms, cloud gaming systems have several significant advantages that attract both game players and developers. In particular, cloud gaming frees players from upgrading their hardware for the latest games since computational hardware is offered by cloud gaming providers, which makes it realistic that a machine with a low-end GPU is able to play graphics-intensive games. Moreover, it allows users to play a game on different platforms, including PCs, laptops, tablets and smart-phones. In

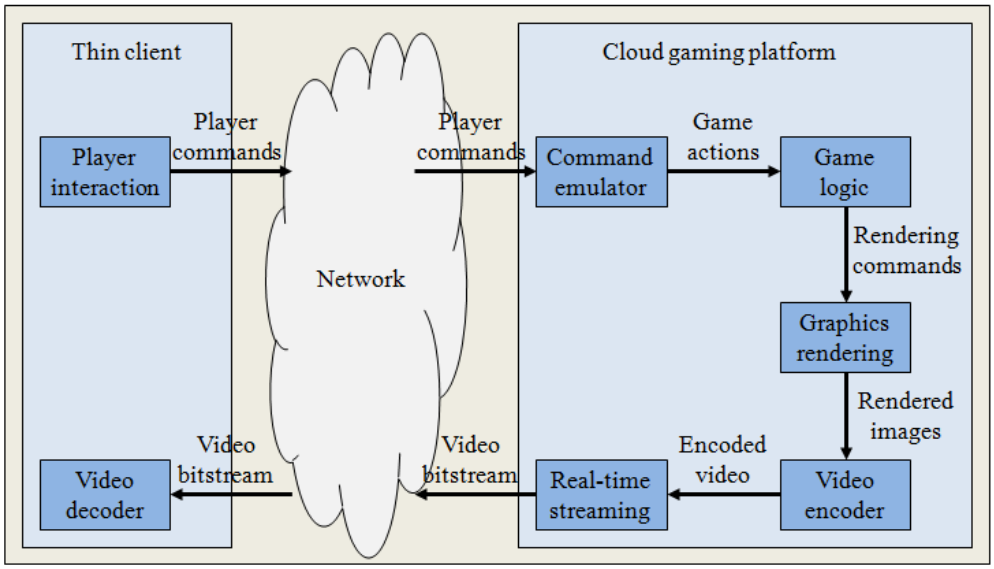


Figure 2.14: The Framework of Cloud Gaming [9]

addition, game developers also benefit from cloud gaming in several aspects. Firstly, cloud gaming offers game developers better digital rights management as the code of cloud games is not directly executed on the player's local device. This eliminates the copyright infringement issues. Secondly, cloud gaming helps game developers solve hardware/software incompatibility issues. Normally, it requires significant effort and resources for developers to achieve compatibility with different operating systems and platforms for the games they developed. Additionally, developers are often required to maintain older versions of games they released, which also consumes a lot of time and resources. Cloud gaming has the advantage of solving compatibility and maintenance issues as game vendors only need to maintain the game software at cloud servers, which requires less resources and makes development of games more cost effective. Thirdly, cloud gaming offers tremendous market potential to game developers. In fact, the cloud gaming market has been the fastest growth component of the game market. Market research predicts that the market of cloud gaming is going to grow to $81 billion by 2017.[19] Additionally, Onlive and Gaikai, two industrial pioneers of cloud gaming systems, have succeeded in gaining multiple millions of users.

2.3.2 Issues and Challenges of Cloud Gaming

Although cloud gaming shows great advantages for both game developers and players, it is still in the early stage as some significant challenges remain regarding the widespread deployment. As low-latency live video streaming and high-performance 3D rendering are key factors to ensure the success of cloud gaming, two performance characteristics, low response delay and high video quality must be ensured in cloud gaming.

While running cloud games, a cloud gaming system has to collect commands from players, process them,

[19] Online Sales Expected to Pass Retail Software Sales in 2013. Website:http://www.neogaf.com/forum/showthread.php?t=444009. Accessed: Dec 02, 2015.

then encode, compress and stream results (game scenes) back to players. To ensure the interactivity, all of these operations must be finished within hundreds of milliseconds [10]. The latency caused by these operations is called response delay, which can be separated into three components:

- **Processing Delay** (PD): the time required for the game server to receive a player's command, process it, and send corresponding encoding scenes back to the player.
- **Playout delay** (OD): The time required for a client to receive, decode and render a frame on the screen.
- **Network Delay** (ND): The time required for a round of data exchange between the server and client. It is usually referred to as the network round-trip time (RTT).

Studies of traditional gaming systems have concluded that different game genres have different thresholds of response delay [10]. Table 2.3 summarizes the maximum delays that a player can tolerate for three popular game categories. Although the traditional server/client gaming platform's response delay tolerance is very similar to cloud gaming's, there is a critical distinction between these two types of game platforms. Traditional online gaming systems often reduce response delay by pre-rendering an action partially on the player's local system before a corresponding command is processed on the game server. However, a thin client in cloud gaming is unable to reduce response delay by adopting such a technique since none of the commands can be handled by the client.

Table 2.3: The Summary of Response Delay on Cloud Games [10]

Model	Examples Genres	Sensitivity	Thresholds (ms)
Avatar - First Person	First Person Shooter (FPS), Racing	High	100
Avatar - Third Person	Sports, Role Playing Games (RPG)	Medium	500
Omnipresent	Real Time Strategy (RTS)	Low	1,000

Another key performance characteristic of cloud gaming is image quality, which refers to the measurement of perceived video degradation at the client side, compared to the original game at the server side. As gaming scenes are encoded/compressed and decoded/uncompressed at the game server and thin client respectively, there might be some loss of data caused by encoder/decoder and network packet loss, which degrades image quality. It is critical to select an excellent video encoder/decoder for cloud gaming as it must quickly encode/compress incoming image frames and distribute them to end users. Currently, the H.264/MPEG-4 AVC [35] encoder is adopted by two main cloud gaming vendors, Onlive and Gaikai, as it has a high compression ratio and can work well with stringent real-time demands. Cloud gaming must consider network conditions while handling video streaming and encoding since some network factors, including delay, jitter, packet loss and packet re-ordering might affect image quality. All these factors might result in video frames not being rendered in time. These factors must be considered while developing a cloud gaming platform.

2.3.3 Existing Cloud Gaming Systems

Cloud gaming has gained a great deal of interest from the industry and several companies have provided or claimed to offer cloud gaming services. Gaikai,[20] which was founded in 2008 and acquired by Sony Computer Entertainment in 2012, is one of the earliest pioneers to offer cloud gaming services. It has developed a high quality, fast interactive cloud-streaming platform, which is capable of quickly delivering games and other interactive content to customers throughout the Internet. One drawback of Gaikai, however, is it does not support use of devices like digital TVs or tablets as client devices. Onlive Game Service,[21] which was released by Onlive corporation in June 2010, is another commercial cloud gaming product. It makes graphics-rich interactive applications available across connected devices such as PCs, laptops and tablets. Additionally, the H.264 encoder is configured to be capable of grabbing the high resolution video frames from each GPU running on their servers. Moreover, it utilizes virtual machines on custom-made servers with GPUs to offer two streams, the live stream and the media stream for each game. The live stream is optimized for real-world game-play, while the media stream is used by players to record and review sequences of their games. StreamMyGame[22] is a similar system to Onlive. It is a software-only game streaming solution which enables Microsoft Windows-based games to be played remotely on Windows and Linux devices. Both these systems have gained success with multiple millions of users. Nevertheless, as they are both closed systems, the architecture and design of these systems are not available to be studied.

Instead of developing commercial cloud gaming platforms, Nvidia released two cloud gaming graphics boards, GRID K340 and 520, and a software development kit called GRID SDK to help build cloud gaming systems. GRID SDK enables fast capture and compression of desktop display or render targets from Nvidia cloud gaming graphics cards or virtual GPUs. As shown in Figure 2.15, GRID SDK mainly consists of two components, NVIFR and NVFBC. NVIFR captures and optionally H.264 encodes from a specific render target, while NVFBC captures and optionally H.264 encodes the entire visible desktop. All generated frames are sent back to remote applications in raw format or H.264 format. As shown in Table 2.4, the K340 is equipped with four entry-level Kepler GPUs, each of which has 384 CUDA cores and 1 GB DDR-5 video memory, while the K520 is configured with two high-end Kepler GPUs, each of which has 1536 CUDA cores and 4 GB of video memory. Both of these GPUs are used to stream consumer games. The GRID K340 is designed for high-density gaming scenarios, and supports the maximum number of concurrent users with high performance of simultaneous encoding. The GRID K520 is designed to be a high performance GPU for high-performance gaming scenarios. The GRID K520 has been chosen as a standard component in Amazon EC2[23] G2 instances, which are Amazon Elastic Compute Cloud (EC2) instances designed for applications that require 3D graphics capabilities. A customized cloud gaming platform can be built using this instance type and the GRID SDK.

[20] Gaikai Official Site. Website: https://www.gaikai.com/. Accessed: Dec 03, 2015.
[21] Onlive Official Site. Website: https://www.onlive.com/. Accessed: Dec 03, 2015.
[22] StreamMyGame Official Site. Website: http://streammygame.com/. Accessed: Dec 03, 2015.
[23] Amazon EC2 Official Site. Website: http://aws.amazon.com/ec2/. Accessed: Dec 03, 2015.

Table 2.4: Main Specifications for Nvidia GRID K340 and K520 [12]

Product Name	GRID K340	GRID K520
Target Market	High-Density Gaming	High-Performance Gaming
Concurrent # Users	4 - 24	2 - 16
Number of GPUs	4 GK107 GPUs	2 GK104 GPUs
CUDA Cores	1536 (384/GPU)	3072 (1536 / GPU)
Memory Size	4 GB GDDR5 (1 GB / GPU)	8 GB GDDR5 (4 GB / GPU)

Cloud gaming has also been well studied in the literature, and some open-source cloud gaming systems have been proposed. These systems can be divided into three categories: (1) 3D graphics streaming, (2) file streaming, and (3) video streaming. The main difference between these three approaches is how they allocate workload for the game server and the thin client.

The early attempts at cloud gaming systems [17] [27] adopted a 3D graphics streaming approach for delivering data between the game server and thin client. With this approach, the cloud game server is only responsible for intercepting, compressing and sending graphics commands to the thin client, while the thin client has to render game scenes using local GPU resources. Therefore, the physical GPU in the thin client must be powerful enough to render game scenes with the guarantee of high quality and real-time. This approach allows the game server to handle more clients as it only has a lightweight workload. Nevertheless, as such a method assigns almost all workload to the thin client, it is less suitable for low-scale devices and less attractive to users.

Similar to 3D graphics streaming, the thin client also has to run games locally in the file streaming approach. Once a game has been selected, a small portion of code is downloaded onto the local device, which allows the player to begin playing the game immediately. Then, the rest of the code is downloaded quickly while the game is being played. Additionally, compute-intensive tasks can be offloaded to the cloud for providing smooth game-play. Recently, file steaming services are becoming popular due to their scalability and affordability. For example, Kalydo gaming Cloud[24] provides smart and extensible toolkits for end users to customize their gaming environment, and has served 50 million sessions in over fifteen countries.

In the video streaming approach, the cloud server processes game commands received from thin clients, makes changed game scenes into 2D videos, encodes and compresses the videos, and streams them to the thin client. The thin client in such an approach only needs to decode, uncompress and display video streams. This approach is ideal for resource-constrained devices and frees users from computation-intensive 3D graphics rendering. Moreover, as video streaming is independent of GPU, the thin client can be easily ported to different platforms, including those with low-capability GPUs such as tablets and smart-phones. One representative cloud gaming system developed by the research community is GamingAnywhere [23] [24], with the

[24]Kalydo Official Site. Website: http://kalydo.com/. Accessed: Dec 05, 2015.

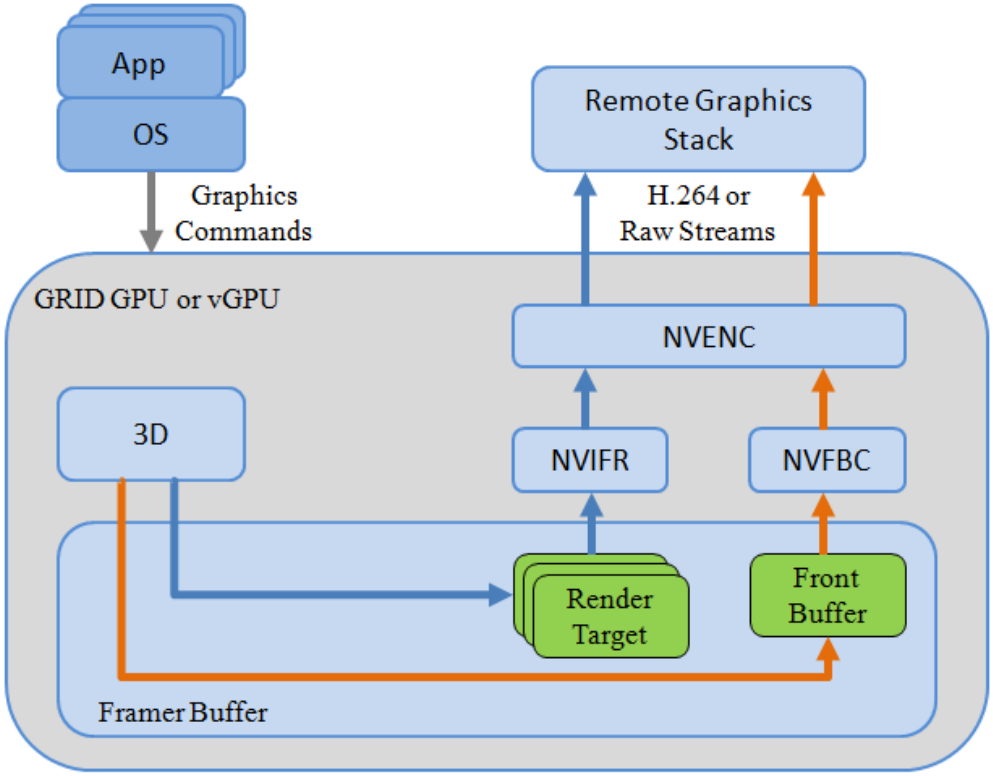


Figure 2.15: The Architecture of GRID SDK [12]

architecture depicted in Figure 2.16. There are two types of flows in GamingAnywhere: data flow and control flow. Data flow is used to stream audio and video frames from game servers to thin clients, while control flow is used to send player actions from thin clients to game servers. Upon the receipt of control messages from a user, the game server represents the user to play a game according to the received commands, and then streams game scenes back. In this case, the GA server is responsible for handling all graphics transactions, while the GA client only has to render game scenes generated by the GA server. GamingAnywhere is an open-source cloud gaming platform with characteristics of high extensibility, portability and reconfigurability. Currently, it supports Windows, Linux, and OS X, and can be ported to iPhone and Android. Thanks to its openness, service vendors and researchers can customize GamingAnywhere to meet their special needs.

2.3.4 Performance Studies of Cloud Gaming

Recently, the potential and performance of cloud gaming has been studied in the literature. Lee *et al.* [30] examined the feasibility of three popular game categories, namely, first person shooter (FPS), role playing games (RPG) and action games in cloud gaming systems. The evaluation they conducted showed some game genres, like FPS, which require stringently low response delay, are not realistic to be deployed in current cloud gaming systems. The survey paper conducted by Shea [37] summarized framework design, issues and challenges of cloud gaming. They also measured the performance of Onlive with different types of games in

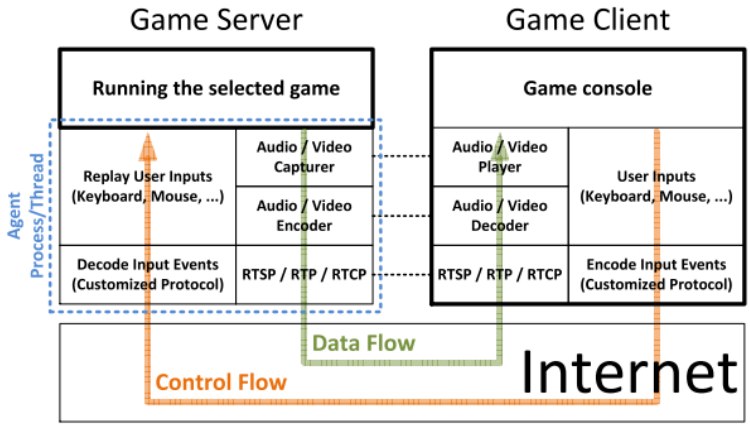


Figure 2.16: A Modular View of GamingAnywhere [24]

terms of interaction delay and streaming quality. The results revealed the potential of cloud gaming, as well as critical challenges regarding the widespread deployment. In particular, they found that the Onlive system managed to keep its interaction delay below 200 ms when network delay was 50 ms. However, when network latency exceeded 50 ms, interaction latency might hinder the user experience. Additionally, image quality dropped sharply when network download bandwidth was below 10 Mb/s. Claypool [11] presented a detailed study of network characteristics of Onlive. They accurately measured Onlive game traffic for several game genres. They analyzed the bitrates, packet sizes and inter-packet times for both upstream and downstream game traffic, and compared them with traditional games and streaming videos. Results showed that downstream and upstream game traffic of Onlive were significantly different that of live videos and traditional game' respectively. Such results are helpful to build effective traffic models for cloud gaming. Cheng *et al.* [5] presented a methodology for quantifying the performance of thin clients in cloud gaming. A demonstration study was performed in this work by using three popular thin-clients, LogMeIn,[25] TeamViewer[26] and UltraVNC[27] to run a classic game, Ms. Pac-Man.[28] The corresponding results showed that display frame rate and frame distortion were both important to games, and different thin-clients showed different levels of robustness against network impairments. Jarschel *et al.* [25] presented a subjective user study to measure user-perceived quality of experience (QoE) in cloud gaming. They defined a set of tests for users to obtain QoE and derived key influence factors and influences of game contents from those QoE results. They determined that user-perceived game experience is not only affected by network delay and packet loss, but also is related to game contents. In particular, the slower the game-play gets, the better QoE is obtained. According to the measurement results, omnipresent perspective games (slow-paced game-play) and third perspective games (medium-paced game-play) are able to gain better QoE than FPS games (fast-paced game-play).

Measuring response delay for thin client games also has been the subject of research. Claypool *et al.*

[25]LogMeIn Official Site. Website: https://secure.logmein.com/. Accessed: Dec 05, 2015.
[26]TeamViewer Official Site. Website: http://www.teamviewer.com/. Accessed: Oct 17, 2014.
[27]UltraVNC Official Site. Website: http://www.uvnc.com/. Accessed: Oct 17, 2014.
[28]Pac-Man Official Site. Website.: http://www.freepacman.org/welcome.php. Accessed: Dec 05, 2015.

[10] performed a study to clarify the impact of Internet latency on different online games. They considered three popular game genres (Avatar Model - First Person, Avatar Model - Third Person and Omnipresent Model) and chose two games for each genre and measured the performance of these games under different Internet latencies. Then they summarized the performance degradation for different classes of online games by depicting an exponential curve fit to the measured data. Based on the summary, they summarized the thresholds of response delay that can be tolerated in three popular game genres. As shown in Table 2.3, the maximum response delay for those games that adopt the Avatar-First Person model like FPS games and racing games is 100 ms. Those games based on the Avatar-Third Person model, like RPGs, normally can tolerate at most 500 ms. Those that use the Omnipresent model such as RTS games, are not strict with response delay, so the maximum response delay that they can tolerate is 1000 ms. Choy [8] performed a measurement study on cloud in terms of end-user latency. In particular, they performed a large-scale latency measurement study from their lab and Amazon EC2 to more than 2500 end-users in the US to determine the percentage of users who can receive tolerable latency while playing cloud games. They found Amazon EC2 was unable to provide acceptable latency for many end-users. Further investigation indicated that the user coverage significantly increased when servers located near the end-users were deployed.

Chen *et al.* [7] proposed a method to measure response delay in cloud gaming systems. The proposed method was based on the fact that most games support a hot key to access the main menu. The key is usually the ESC key for computer games and the START button for console games. Based on the assumption that the ESC key is the hot key for invoking the main menu, the time difference between pressing the ESC key and rendering the first frame of the main menu at the server side is the response delay. Therefore, they utilized a hooking mechanism in the Windows operating system to capture the "ESC pressed" event and monitored the game screen to obtain the time when the main menu was on display. The proposed methodology can be widely applied as it does not require openness of systems and thus can be plugged into any closed systems. Additionally, two well-known cloud gaming systems, Onlive and StreamMyGame, were evaluated using this proposed method. The experimental results showed that Onlive achieved acceptable response delay, whereas StreamMyGame suffered from a high response delay that players could not tolerate. Based on this work, Chen *et al.* [6] further proposed a suite of measurement techniques to evaluate the quality of service (QoS) of cloud gaming systems including response delay measurement, game delay measurement, network traffic analysis and image quality measurement under real-world network conditions. Then they used their methods to compare two commercial cloud gaming systems: Onlive and StreamMyGame. The measurement results showed that compared to StreamMyGame, Onlive provided adaptable frame rates, better image quality, and shorter server game processing delays, but consumed less network bandwidth.

There also has been some research work related to measuring video quality in cloud computing. Wang *et al.* [45] surveyed the existing proposals for measuring video quality including Peak-Signal-to-Noise-Ratio (PSNR) [46] and Structural Similarity Index (SSIM) [47], with respect to their advantages and disadvantages. A PSNR value over 40 decibels (dB) typically indicates an excellent image that is very close to the original

one [46], thus proving that the reconstructed game video is very good. A value between 30 to 40 dB means a good image, between 20 and 30 dB is quite poor. Additionally, a value lower than 20 dB is unacceptable. In SSIM, the value closer to 1 is regarded as a better result.

The research community is also actively resolving the issues of virtualized GPU isolation and scheduling in cloud gaming. Yu *et al.* [48] proposeed VGRIS, a scheduling framework for virtualized GPU resource isolation and scheduling, which enables a single GPU to be shared efficiently by different VMs composed of VMware and virtualBox in cloud gaming. It adopted an API interception library to manage underlying resource scheduling for multiple VMs. Only a few binaries within the intercepted library need to be modified. Based on the VGRIS framework, they implemented three scheduling algorithms for addressing various performance requirements: high performance of Service-level Agreement[29] (SLA) proportional resource scheduling, and performance and fairness trade-off. Their experimental results showed that the overhead of the VGRIS framework (extra execution time of testing benchmarks) was limited to 3.59%, and that this framework could effectively schedule GPU resources on more than one virtualization platform simultaneously. Similarly, Zhang *et al.* [49] presented VGASA, an adaptive scheduling algorithm for virtualized GPU resources in cloud gaming. VGASA is realized by leveraging vSGA technology in VMware and API interception technology. The basic idea of VGASA is to collect feedback from each VM and schedule GPU resources based on the scheduling algorithms they realized. They implemented three algorithms for achieving different goals. More specifically, SLA-Aware aimed to achieve the SLA requirement for each VM, Fair SLA-Aware aimed to maximize the usage of GPU resources by reallocating GPU resources from VMs with higher frame rates to those who do not meet SLA requirements, while Enhanced SLA-Aware balanced the gaming performance and number of users on a single GPU. Compared to VGRIS, one benefit of VGASA is that the scheduling algorithms they realized are adaptive in response to uncertainties of running time. Meanwhile, their experimental results showed that the GPU performance overhead of VGASA is limited to 5-12% of the benchmark execution time. Hong *et al.* [21] conducted measurement studies to derive the game-dependent parameters for QoE and for a performance model. Based on these factors, they proposed a heuristic algorithm, called quality-driven heuristic (QDH). QDH is able to consolidate more VMs on a server as long as the user-specified maximal tolerable QoE degradation is not exceeded. They utilized GamingAnywhere and VMware ESXi 5.1 to build a prototype implementation, to evaluate the efficiency of the proposed algorithm. Their simulation results indicated that QDH resulted in close-to optimal performance. Additionally, they also found that QDH was able to scale to large cloud gaming services with 20,000 servers and more than 40,000 gamers. Hong *et al.* [22] conducted detailed experiments using modern GPUs and GamingAnywhere to answer the question: are modern GPUs ready for cloud gaming? In particular, they compared the performance of GPU pass-through and vGPU technology, running the same benchmarks on GPU pass-through VM instances and vGPU VM instances. Through the experiments, they found that 1) virtualized GPUs outperformed pass-through GPUs

[29] A service-level agreement is a contract between a service provider and its customers that documents what services the provider will furnish and with what performance guarantees.

in some benchmarks, especially 2D-intensive benchmarks and games, while pass-through GPUs produced better results in 3D-intensive benchmarks than virtualized GPUs, and 2) shared virtualized GPUs were reasonably scalable with respect to the number of VMs and were able to achieve stable performance.

Chapter 3

Experimental Setup

This chapter describes the context of the evaluation, the experimental design as well as the general and specific approach to measurement and comparison of tested GPU virtualization techniques. Section 3.1 describes the graphics card benchmarks, gaming applications and performance monitoring tools used to evaluate the GPU virtualization performance. Section 3.2 contains the description of the performance metrics that are used in the experiments, while Section 3.3 discusses the tested scenarios and hardware configuration in the experiments. Additionally, descriptions of the measurement techniques are provided in Section 3.4.

3.1 Experimental Tools

This section introduces all of the experimental tools used in the experiments, including the graphics card benchmarks, gaming applications, VMware Horizon View, GamingAnywhere, and performance monitoring tools. The graphics card benchmarks and gaming applications run on both virtualization and non-virtualization environments for evaluating performance of each tested configuration. VMware Horizon View is used as a cloud gaming platform, while GamingAnywhere is used to capture system times and images which are used to calculate response delay and image quality. Performance monitoring tools are responsible for capturing time stamps of hardware events invoked by benchmarks and gaming applications.

3.1.1 Graphics Card Benchmarks

Graphics card benchmarks run a set of GPU-intensive programs to evaluate the performance of a given GPU. They can be used objectively to measure many aspects of GPU performance such as capability in terms of frames per second. Graphics card benchmarks focus on the application-level performance. Additionally, graphics card testing includes synthetic tests and real-world tests. Synthetic tests are purely designed for stressing the hardware to its fullest potential. Although it is not demonstrative of real-world scenarios, a synthetic test is usually the most useful in determining the maximum possible performance of a graphics card (its workload capacity in terms of frames per second). Unlike a synthetic test, a real-world test is based on real games and used to evaluate GPU performance in real-world scenarios. For evaluating GPU performance in each tested configuration, several graphics card benchmarks, including two synthetic tests and a real-world gaming test, are used in the experiments. Those benchmarks are the following and their features are shown

in Table 3.1.

- **Unigine Sanctuary Benchmark**[1] is a GPU-intensive benchmark for GPU stress testing, which stresses a graphics card to its limits. It adopts Unigine Engine to provide rich graphics by leveraging the most advanced capabilities of graphics APIs (DirectX/OpenGL). While running the benchmark, it renders a gothic chapel with still statues lit by the light of torches. This tool can be effectively used to determine the stability of a GPU under extremely stressful conditions. It provides objective results and generates in-game rendering workloads across all platforms, including Windows, Linux and Mac OS X.
- **PassMark PerformanceTest**[2] is a tool that can provide objective benchmark results on a PC using a variety of different speed tests, including the tests for CPU, memory, hard disk and graphics card. In terms of graphics card tests, it provides a series of basic 2D tests which draw lines, bitmaps, fonts, text, and GUI elements to evaluate the 2D features of a graphics card, including simple vector, complex vector, fonts and text rendering, image filtering functionality, image rendering and DirectX 2D computing capability. It also provides an advanced 3D graphics card test to benchmark how well a graphics card performs by using the most common features of DirectX. The test renders different scenes in windowed or full screen mode according to the selected DirectX API. For instance, it will render a scene with three fighter planes flying over several sea islands if you choose DirectX 9 as rendering API. In addition, it also offers a point-based ranking system to help users compare their graphics cards' performance.
- **Doom 3**[3] is a horror first person shooter computer game released in 2005. It utilizes DirectX 9 to provide high-quality graphics features. Moreover, it provides a time demo benchmark to help users measure frame rate (in frames per second) that graphics cards can generate in Doom 3. This time demo benchmark mainly renders preset frames, in which a person used guns to fight against lots of monsters, to test how fast a graphics card can handle those frames.

3.1.2 Performance Monitoring Tools

Although we can gain performance insights though simple performance metrics, such as frames per second (fps), from graphics card benchmarks, it is still necessary to make further resource utilization analysis to determine the main bottleneck. Therefore, the following two performance monitoring tools are chosen to keep track of hardware resource utilization of each graphics card benchmark. These tools are the following:

- **MSI Afterburner**[4] is a free utility which is compatible with almost all graphics cards. It enables users to monitor all kinds of critical hardware resource information such as CPU usage, GPU usage, and graphics memory usage in real time. Additionally, it also provides logging functionality that enables users to record all hardware resource utilization periodically. Therefore, running graphics card benchmarks or gaming

[1]Unigine Sanctuary Benchmark Official Site. Website: https://unigine.com/products/sanctuary/. Accessed: Dec 10, 2015.
[2]PassMark PerformanceTest Official Site. Website: http://www.passmark.com/products/pt.htm. Accessed: Dec 10, 2015.
[3]Doom 3 Official Site. Website: http://bethsoft.com/en-us/games/doom_3_bfg. Accessed: Dec 10, 2015.
[4]MSI Afterburner Download Official Site. Website: http://event.msi.com/vga/afterburner/. Accessed: Dec 10, 2015.

Table 3.1: The Features of the Tested Benchmarks

Benchmarks	Features
Unigine Sacntuary	DirectX benchmark; Extreme hardware stability testing; Multi-Platform support for Windows, Linux and Mac OS X; Support for DirectX 9, DirectX 11 and OpenGL 4.0; Highly customizable configuration.
PassMark PerformanceTest	DirectX benchmark; Comprehensive test for GPU in terms of 2D and 3D capability; Support for DirectX 9 and higher; Has a point-based ranking system.
Doom 3	DirectX benchmark; Provides real gaming scenarios tests; Support for DirectX 9 and higher.

applications alongside this tool makes it easy to do the full analysis of GPU utilization and other hardware resource usages.

- **GPU-Z**[5] is a lightweight system utility designed to provide vital information of graphics cards. This tool displays all the important information, such as graphics memory type, memory size, bus bandwidth and supported computing type. It also provides monitoring functionality to monitor GPU resource usages in real time.

3.1.3 VMware Horizon View

VMware Horizon View is a desktop virtualization solution that delivers virtual Windows desktops and applications to end users so they can work anytime, anywhere, on any device. With the help of VMware Horizon View, you can simplify and automate the management of virtual desktops, and support users with access to all their Windows desktops and online resources through a unified workspace [2]. Moreover, VMware Horizon View enables end users to utilize VMware's GPU virtualization techniques, including vSGA, vDGA and vGPU in their virtual desktops. VMware Horizon View includes six main components, and they are briefly introduced as follows:

- **View Connection Server** simplifies the management and deployment of virtual desktops. Administrators can centrally manage their virtual desktops through a single console provided by this server, while end users access their personalized virtual desktops through this server.

[5]GPU-Z Official Introduction Site. Website: https://www.techpowerup.com/gpuz/. Accessed: Dec 10, 2015.

- **View Security Server** is a server that adds an additional layer of security between your internal network and the Internet. With this server, users can only access the virtual desktops for which they are authorized.
- **View Composer Server** provides a service to create clone desktops, by creating master images that share a common virtual disk. These images are one or more copies of the image of a parent virtual machine. They operate as individual virtual machines but share the virtual disks of the parent.
- **Horizon Client** provides the connection to remote virtual desktops from end users' devices. VMware Horizon Client is available now for Windows, Ubuntu Linux, Mac, iPhone and Android.
- **View Persona Management** is an optional component that provides dynamic user profiles across user sessions on different desktops. With this component, end users can use and maintain their designated settings between sessions.
- **View Agent** takes charge of the communication between virtual machines and VMware Horizon View Client. This component must be installed on all virtual machines so that View Connection Server can communicate with them. This component also provides some useful features such as connection monitoring, virtual printing and access to locally connected USB devices. Additionally, VMware Inc. provides a plug-in component called View Agent Direct-Connection for any Horizon Client to connect directly to a virtual desktop without using View Connection Server. This plug-in component provides the flexibility of directly accessing virtual desktops without installing the components described above.

Figure 3.1 illustrates the VM setup with VMware Horizon View for deploying/running graphics card benchmarks and cloud games. Refer to Table 3.3 for descriptions of the experimental machines and to Figure 3.3 for the network topology. View Agent Direct-Connection is used in the experiments so that physical machines (Machine 2 or Machine 3) can directly launch VMware's virtual desktops with vSGA, vDGA or vGPU enabled, through Horizon Client without using View Connection Server. Tested graphics card benchmarks and cloud games run on virtual desktops. GA client triggers the game events and GA Server captures system times and frame samples during the execution of tested games.

3.1.4 GamingAnywhere

GamingAnywhere [23] is an open-source cloud gaming platform that can be used to deploy cloud games and study cloud gaming due to its openness. Figure 3.2 illustrates a sample cloud gaming service based on GA. A player creates commands from the mouse, keyboard and touch input and submits them to a game server. The game server uses the received commands to play the game and streams encoded frames of game screens to the client. Finally, the game client receives, decodes and renders game frames to the local console. In the experiments, however, as VMware Horizon View has already acted as cloud gaming platform, GamingAnywhere is not utilized to deploy cloud games but to capture system times and images, which are then used to calculate response delay and image quality respectively. The concrete methods of measuring response delay and image quality are explained in Section 3.4.

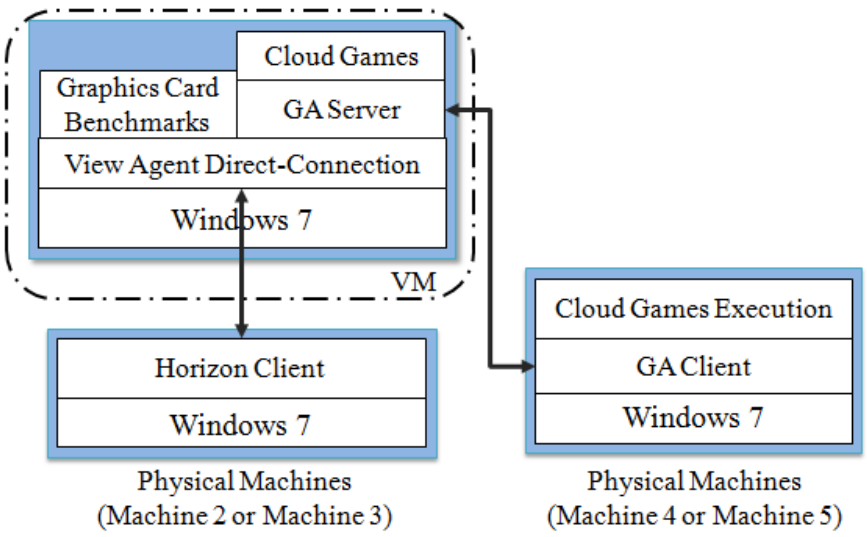


Figure 3.1: VM Setup with VMware Horizon View

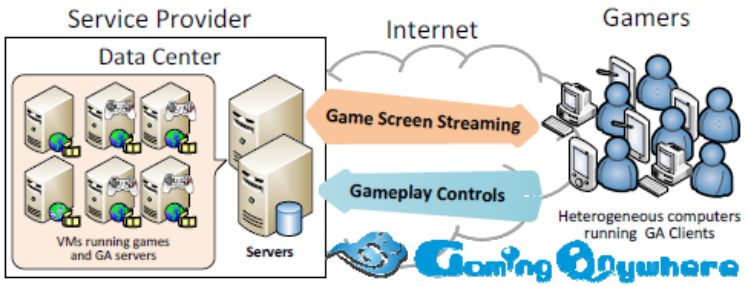


Figure 3.2: A Cloud Gaming Service Using GamingAnywhere[24]

3.1.5 Cloud Games in the Experiments

Since the performance of cloud gaming systems may be game-dependent, it is necessary to measure each game category's impact. Three game categories are considered in the experiments [10]: Avatar-First person, Avatar-Third person and Omnipresent. A representative game for each category is selected, and they are briefly introduced as follows.

- **AssaultCube**[6] is a free, multi-player, first person shooter (FPS) game. Combat scenarios in the game are designed to be as close to those in real scenes as possible. It is a small game with size of only about 40 MB and is available for Windows, Mac and Linux.
- **LEGO Batman 2: Super Hero**[7] is an action-adventure game developed by Travellers Tales in 2008 that follows the mode of Avatar-Third person. It also can be viewed as a role playing game (RPG). In this game, all interactive objects are made of LEGO bricks, and a player controls his character to fight against enemies from a third person perspective.

[6]AssaultCube Website. Website: http://assault.cubers.net/. Accessed: Dec 10, 2015.
[7]LEGO Batman: The Videogame. Website: http://games.kidswb.com/official-site/lego-batman/. Accessed: Dec 10, 2015.

- **Warhammer 40,000: Dawn of War II**[8] is a real-time strategy (RTS) game developed by Relic Entertainment in 2009 that adopts the Omnipresent model. In the game, a player controls his squad to fight against enemies and destroy the enemies' buildings in order to get the victory.

3.1.6 Discussion About Experimental Tools

Unused Virtualization Techniques

In the initial experimental design, the virtualization solutions provided by another GPU virtualization product, Citrix's Xenserver, were also going to be evaluated in the experiments. Nevertheless, these solutions were abandoned for the following reason: In XenServer, a license server is utilized to manage the license information for all XenServer products. This server is required provided you want to use advanced features of XenServer such as vGPU technology. In the experiments, a VM is created to act as license server. However, this server cannot be connected and communicated to even though several methods to fix this issue have been tried. Consequently, the advanced features of XenServer such as vGPU technology cannot be accessed as the license of XenServer cannot be updated to the license server. Therefore, the plan of evaluating GPU virtualization techniques in Citrix's XenServer was abandoned.

The Usage of GamingAnywhere

As well, GamingAnywhere was intended to be utilized as a cloud gaming platform for deploying cloud games. Nevertheless, in the experiments, GA is only utilized to capture frame samples and system times to measure image quality and response delay. This change is made for the following reason: In the experiments, VMware Horizon View is used to launch GPU virtualization instances. When cloud games run on these instances, VMware Horizon View actually acts as a cloud gaming platform. Therefore, it is not necessary to use GA as a cloud gaming platform. Additionally, while running as a cloud gaming platform, GamingAnywhere introduces extra hardware consumption. This is because the GA server must be launched in a physical machine and the GA client runs on another physical machine. In contrast, when utilizing VMware Horizon View as a cloud platform, only one physical machine is required. For these reasons, GamingAnywhere is used as an assistant program to capture frame samples and system times, instead of as a cloud gaming platform.

3.2 Performance Metrics

Table 3.2 shows the performance metrics that are used in each benchmark/gaming application used in the experiments. As illustrated in this table, the benchmarks and gaming applications described above have different associated performance metrics. There are several metrics that need to be considered:

[8]Warhammer 40,000; Dawn of War II. Website: http://www.dawnofwar2.com/. Accessed: Dec 10, 2015.

Table 3.2: Performance Metrics Used in each Tested Benchmark/Gaming Application

Benchmarks/Gaming Applications	Performance Metrics
Unigine Sanctuary	Frame Rate (fps)
Passmark PerformanceTest	Score Frame Rate (fps)
Doom 3	Frame Rate (fps)
AssaultCube	Image Quality (decibel (dB)) Response Delay (ms)
LEGO Batman 2: Super Hero	Image Quality (decibel (dB)) Response Delay (ms)
Warhammer 40,000: Dawn of War II	Image Quality (decibel (dB)) Response Delay (ms)

- **Frame Rate**, as measured by frames per second, is the frequency at which a graphics card produces unique consecutive images, which to some degree, measures the throughput capability of the given graphics card. As shown in Table 3.2, frame rate is used as a performance metric for PassMark PerformanceTest 3D benchmark, Unigine's Sanctuary benchmark and Doom 3 benchmark.
- **Score** is a simple form of performance metric used by the PassMark PerformanceTest 2D benchmark. The score of a graphics card has no meaning unless being compared to other graphics cards' scores. PassMark PerformanceTest offers a point-based rank system that enables users to compare their benchmark results with others.
- **Image Quality** is one of the most important metrics when evaluating the performance of cloud games. It is the measurement of perceived image degradation at the client side, compared to the original game frame at the server side. Relatively high image quality should be achieved since low image quality turns players away from cloud games. In the experiments, image quality is quantified using the PSNR.
- **Response Delay** is another key metric of cloud gaming performance. It refers to the time difference between when a player creates and submits a control command to a server and when the corresponding frames are rendered on the screen. Different game genres have different requirements of response delay. For instance, a FPS game requires the response delay to be less than 100 milliseconds [10]. Cloud Gaming platforms should ensure that such requirements are met for each game genre.

3.3 Hardware Configuration and Tested Configuration

3.3.1 Hardware Configuration

Table 3.3 shows the machines that are used in the experiments. A modern mid-range server (Machine 1), Dell PowerEdge R730, is used to deploy VMware ESXi. All the VMs are created and managed by this server. It is equipped with an Nvidia GRID K1 graphics card, that supports all VMware GPU virtualization solutions. The graphics card consists of four entry-level Kepler GPUs configured with 192 CUDA cores and 4 GB DDR-3 graphics memory. Machine 2 is used as bare-metal system 1 and VMware Horizon View Client 1. While running as bare-metal system 1, all the results it produces are regarded as baseline results in the first group of experiments. While running as VMware Horizon View Client 1, it launches each tested GPU virtualization instance, including vDGA, vSGA and vGPU instances, and runs the tested graphics card benchmarks and games remotely.[9] Machine 3 is used in a similar way as Machine 2, which performs as bare-metal system 2 and VMware Horizon View Client 2. Machine 4 and Machine 5 act as client machines to run GA client 1 and GA client 2 respectively, which are used to trigger the events of capturing system times and frames for calculating the response delay and image quality of each tested game.

From Table 3.3, we can see that Machine 2 has a faster CPU, more RAM and more hard disk space than Machine 3. In particular, Machine 2 is equipped with a better GPU than Machine 3. Table 3.4 compares the configurations of the GPUs in these two machines, from which we can see the Nvidia NVS 4200M in Machine 2 has higher number of cores and memory speed, and more memory space than Radeon X1300 in Machine 3. As we know, the higher a GPU's core speed is, the faster it can run and produce better performance. Therefore, when running as bare-metal systems, it is expected that bare-metal system 1 (Machine 2) achieves better performance than bare-metal system 2 (Machine 3).

Table 3.5 shows the hardware configuration for each GPU virtualization solution. Each GPU virtualization instance is equipped with 2GB RAM and 100 GB hard disk, but different GPU resources. vSGA VM has only 12 GPU cores and 256 MB graphics memory, while GPU core and memory resources for vGPU K120Q are twice that for vSGA. vGPU K140Q has 48 cores and 1 GB graphics memory, while both vGPU K180Q and vDGA have exactly the same GPU resource. In the experiments, the instances that have more GPU cores are expected to obtain the better results. The configuration of graphic memory for each tested configuration should not affect the experimental results as none graphics -memory-intensive benchmark or gaming application is used in the experiments. Therefore, even the vSGA instance that has only 256 MB of graphics memory meet the requirement of graphics memory of each tested benchmark and gaming application.

Except for those parameters controlled by each benchmark, other parameters of the graphics card in the experiments are set as shown in Table 3.6.[10] Additionally, the parameters only available in some specific

[9]The tested benchmarks run directly on each tested GPU virtualization instance, while the tested games run on top of GamingAnywhere, which is used to capture system times and image samples for calculating response delay and image quality.

[10]This configuration does not apply to the vSGA instance as it does not provide an interface for users to modify the configu-

Table 3.3: Experimental Machines

Experimental Machines	**Machine 1**	**Machine 2**	**Machine 3**	**Machine 4**	**Machine 5**
Machine Usage	VMware ESXi	Bare-metal 1 & VMware Horizon View Client 1	Bare-metal 2 & VMware Horizon View Client 2	GA Client 1	GA Client 2
CPU Info	$2 \times 12 - core$ Intel Xeon @ 2.4GHz	$1 \times dual - core$ Intel Core i5-2520M @ 2.5 GHz	$1 \times dual - core$ Intel Core i5-6600 @ 2.4 GHz	$1 \times dual - core$ Intel Core i5-E6550 @ 2.33 GHz	$1 \times 4 - core$ Intel Core i7-2860QM @ 2.50GHz
GPU Info	1 Nvidia GRID K1 / 4 GB Memory	1 Intel Graphics 3000 / 1 GB Memory & 1 Nvidia NVS 4200M / 1 GB Memory	1 ATI Radeon X1300 / 256 MB Memory	1 Intel Q35 / 256 MB Memory	1 Nvidia GeForce GTX 970M / 3GB Memory & 1 Intel HD 4600 / 1GB Memory
RAM	16 GB DDR-4 @ 1867 MHz	4 GB DDR-3 @ 1333 MHz	1 GB DDR2 @ 800 MHz	1 GB DDR2 @ 667 MHz	16 GB DDR-3L @ 1.6 GHz
Hard Disk	2 TB	500 GB	250 GB	160 GB	1 TB

GPU virtualization instances are disabled for fair comparison.

Figure 3.3 shows the network topology in the experiments. Firstly, a VM is created and used to install vCenter Server. All other VMs are managed via VMware vSphere Web Client in vCenter Server. Secondly, another VM acts as an Active Directory Server, DNS and DHCP Server to manage a local domain and a private network which is used to connect all experimental physical machines and VMs. In order to let all experimental machines and VMs access the Internet, SmallWall, an open-source firewall, is installed on another VM to act as a bridge between public and private network traffic. Furthermore, all the experimental physical machines are joined by a physical switch to form a private network.

3.3.2 Tested Configuration

Figure 3.4 and Figure 3.5 summarize the tested scenarios and corresponding instances in the experiments, and Table 3.5 shows the machine characteristics of each GPU virtualization solution. From the figures and

ration of the graphics card.

Table 3.4: The Comparison of GPUs in Machine 2 and Machine 3

GPUs	Nvidia NVS 4200M	Radeon X1300
GPU Core Speed (MHz)	810	450
Graphics Memory Size (MB)	1024 DDR-3	256 DDR
Graphics Memory Speed (MHz)	800	250

Table 3.5: Hardware Resource Configuration for each GPU Virtualization Instance

GPU Types	vSGA	vGPU K120Q	vGPU K140Q	vGPU K180Q	vDGA
Graphics Memory (MB)	256	512	1024	4096	4096
GPU Cores	12	24	48	192	192
RAM (GB)	2	2	2	2	2
Hard Disk (GB)	100	100	100	100	100

table we can see that all GPU virtualization solutions of VMware, including GPU pass-through (vDGA) and GPU sharing (vSGA, vGPU) are evaluated in the experiments. In particular, as shown in Table 2.2, Nvidia K1 provides multiple vGPU profiles for supporting various use cases. For fully analyzing the performance of the vGPU solution, it is evaluated with three configurations, namely, vGPU K120Q, vGPU K140Q and vGPU K180Q. These three configurations are respectively designed for low-end, medium-end and high-end applications.

The experiments are conducted as follows. Firstly, the performance of the bare-metal system and each instance of GPU virtualization solution is evaluated by running the graphics card benchmarks and cloud games that are introduced in Section 3.1. Secondly, all the results produced by the bare-metal system are treated as baseline data, then the performance improvement/degradation of each GPU virtualization solution is calculated by comparing the results of each GPU virtualization solution with the baseline data. Thirdly, as illustrated in Figure 3.4, in the first group of experiments, single instances of GPU pass-through and GPU sharing solutions are evaluated to assess the performance of each GPU virtualization solution. Fourthly, as shown in Figure 3.5, to evaluate the potential scalability of each solution, double instances of each solution are evaluated in the second group of experiments.

3.4 Mesurement Techniques

3.4.1 Measuring GPU Performance

The graphics card benchmarks mentioned above are utilized to measure GPU performance for both 2D and 3D graphics. The Ungine Sanctuary benchmark runs in three configurations, as shown in Table 3.7, to

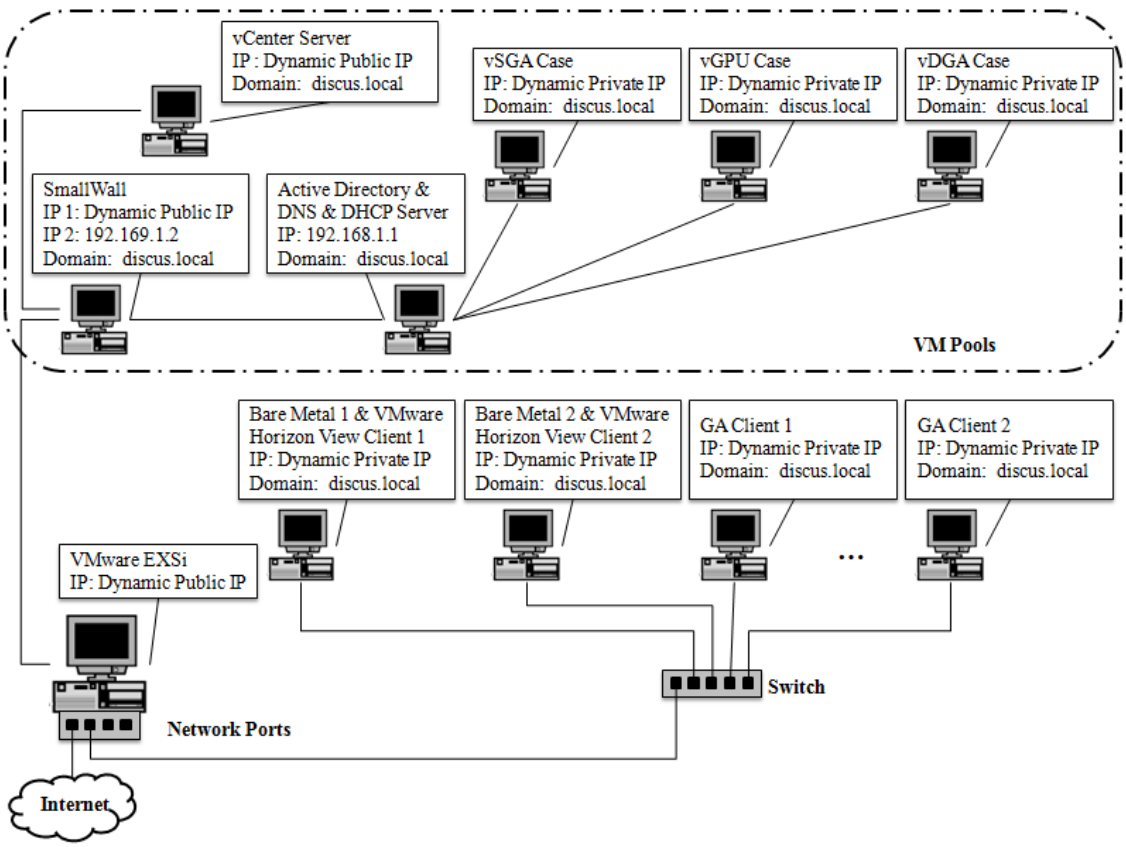


Figure 3.3: Network Topology of Experiments

evaluate GPU performance under various stress conditions. DirectX 9 is selected as vSGA does not support DirectX 10 or higher. The benchmark runs in windowed mode with the resolution 1280×800 as none of the VM types can load it in full screen mode. Ambient occlusion, Shaders, Anisotropy and Anti-aliasing are set differently in different configurations. Ambient occlusion adds realism to scenes via reducing the intensity of ambient light on surfaces. It also enhances depth perception by offering a soft shadow effect for objects. Shaders are used to calculate rendering effects on graphics hardware with a high degree of flexibility. With Shaders, customized effects can be made. The position, saturation, brightness, and contrast of all pixels, vertices, or texture used to construct a final image can be changed by Shaders. Anisotropy affects the crispness of textures, while Anti-aliasing allows users to minimize the visible aliasing on the edges of images with transparent textures.

The PassMark PerformanceTest benchmark is used to test 2D and 3D capability. The 2D tests run with the default configuration to test 2D features of each GPU case in the experiments including simple vector, complex vector, fonts and text rendering, window interface, image filtering, image rendering and DirectX 2D computing. Table 3.8 shows the configurations that are used to run its 3D Simple and Complex tests. These two 3D tests share the same configuration except test types. Anti-aliasing is disabled in this benchmark as it is not supported in the vSGA instance. Vertical Sync can improve image quality by eliminating horizontal

Table 3.6: Configurations of Graphics Card in the Experiments

Parameter	Setting
Antialiasing - FXAA	On
Antialiasing - Gamma correction	On
Buffer - Flipping mode	Use block buffer
Memory allocation policy	Aggressive pre-allocation
Power management modes	Prefer maximum performance
Threaded optimization	On
Triple buffering	On
Virtual Reality pre-rendered frames	4

Table 3.7: Configurations of Unigine Sanctuary Benchmark

Configurations	**Low**	**Medium**	**High**
API	DirectX 9	DirectX 9	DirectX 9
Ambient Occlusion	No	Yes	Yes
Shaders	Low	Medium	High
Anisotropy[11]	1	8	16
Anti-aliasing[12]	Off	4x	8x
Resolution	1280×800	1280×800	1280×800

tearing effects in the 3D image.

As for the Doom 3 benchmark, it runs with the default configuration in all environments. The settings for the main parameters are shown in Table 3.9. The resolution for this benchmark is set to 1280×800, while Anisotropy is set to 4. Anti-aliasing and Vertical Sync are disabled in this benchmark. Additionally, this benchmark pre-caches textures to prevent stuttering/jerkiness and sets the image quality to medium.

3.4.2 Measuring Hardware Consumption

For further analysis of GPU virtualization performance, it is critical to record the usage of hardware resources and GPU events during the execution of benchmarks/gaming applications. For recording hardware resource consumption, MSI Afterburner is launched before the execution of benchmarks/gaming applications with the following settings: 1) polling period of 1 second; 2) logging functionality enabled; and 3) record all key events (such as GPU usage and graphics memory usage, etc.). Then MSI Afterburner is kept running during the execution of benchmarks/gaming applications. Nevertheless, in this case, MSI Afterburner does not just record the hardware resource consumption of the benchmarks/gaming applications, but also that of system programs. For obtaining only the hardware resource consumption of each benchmark/gaming

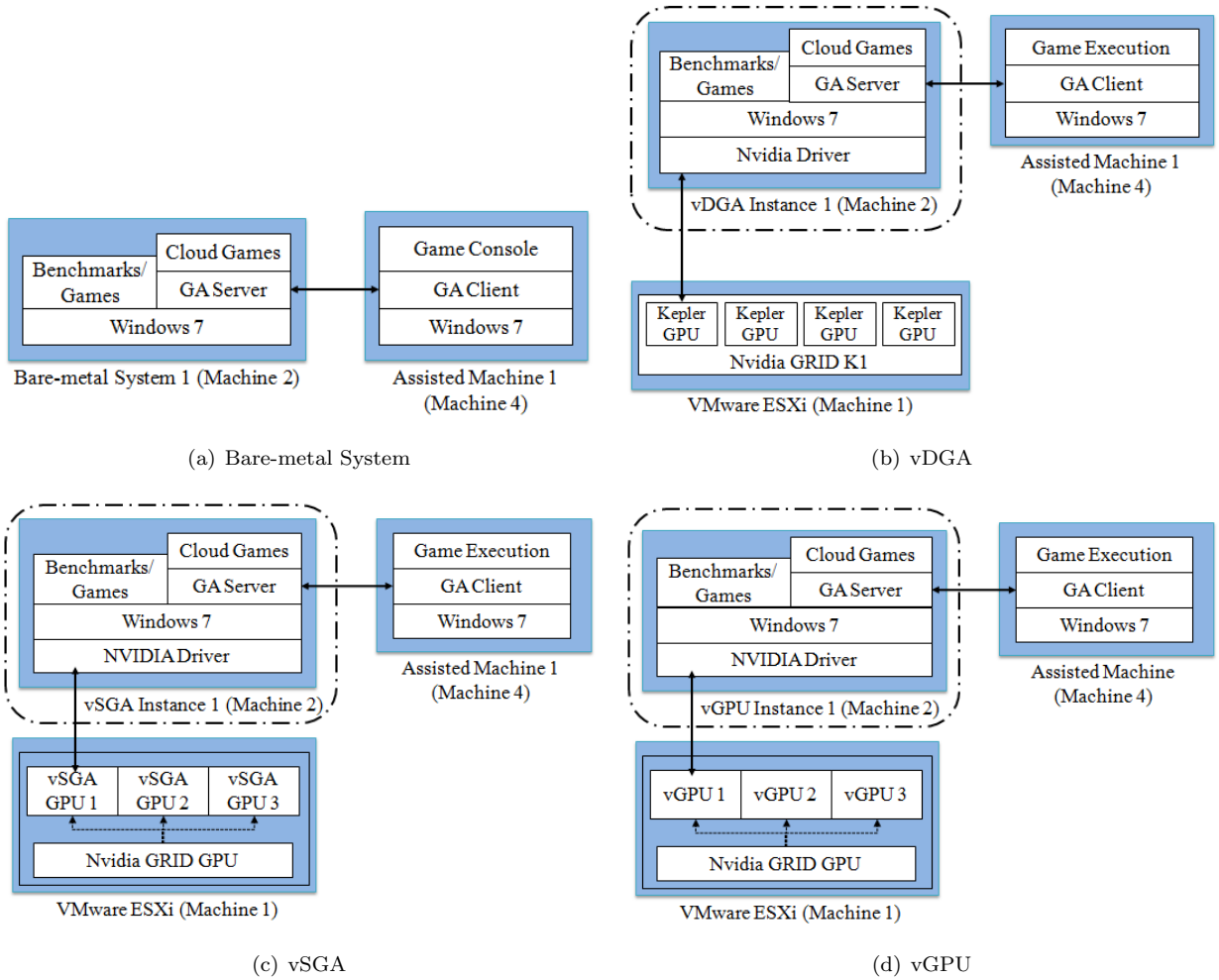


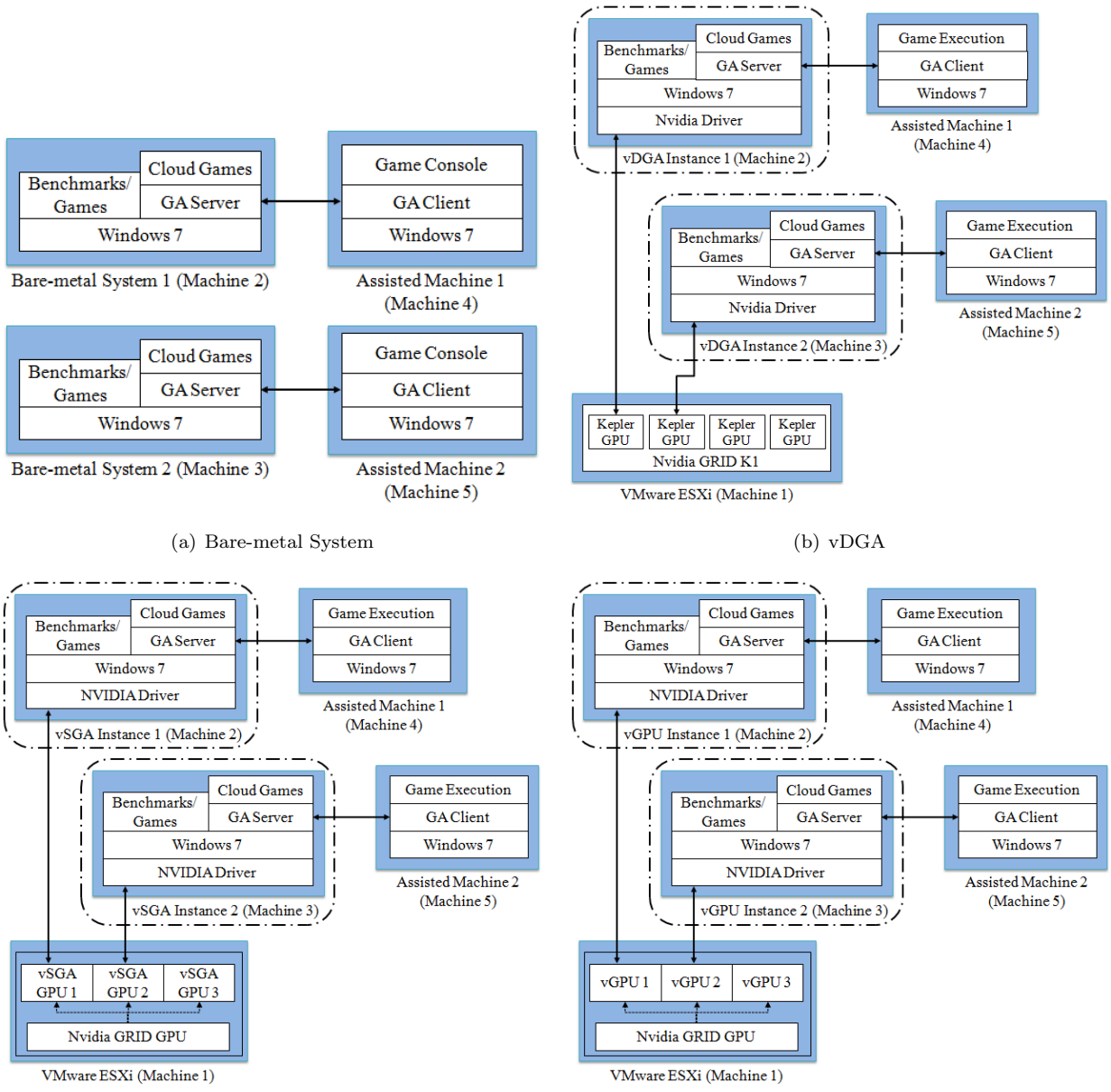
Figure 3.4: Tested Scenarios: Single Instance

application, MSI Afterburner runs with no benchmark/gaming application for 60 seconds to obtain the base hardware resource consumption of system programs. Then the hardware resource consumption of the benchmark/gaming application is calculated using the total consumption minus the base consumption.

GPU-Z is used in a similar way to record GPU usage as MSI Afterburner fails to capture this piece of information in vGPU environments. Therefore, GPU-Z is used to record GPU usage, while MSI Afterburner is utilized to capture other resource usage.

3.4.3 Measuring Response Delay

For measuring the response delay of the tested games running on GamingAnywhere, the measurement method proposed by Chen *et al.* [7] is adopted. Normally a hot key event is utilized in this method. For instance, the ESC key is usually the hot key of invoking a game's main screen. As illustrated in Figure 3.6, assuming the ESC key is pressed at time *t0* and the main menu screen at time *t4*, then the time difference (*t4* -



(a) Bare-metal System

(b) vDGA

(c) vSGA

(d) vGPU

Figure 3.5: Tested Scenarios: Double Instances

Table 3.8: Configurations of Performance 3D Benchmark

Configurations	Simple	Complex
Test Type	DirectX 9 - Simple	DirectX 9 - Complex
Resolution	1024×768	1024×768
Anti-aliasing	Off	Off
Vertical Sync	On	On
Scene Detail	3 planes, 200 trees	3 planes, 200 trees
Test Duration (s)	60	60

Table 3.9: Configurations of Doom3 Benchmark

Resolution	1280×800
image_useCache	Pre-caches textures to eliminate jitter
com_machineSpec	Medium image quality
image_anisotropy[13]	4
r_multiSamples[14]	Off
r_swapInterval [15]	Off

t0) is the response delay of the ESC key. Nevertheless, the time of interest in the thesis experiments is different. As the thesis focuses on the performance of each tested GPU virtualization solution. Therefore, as illustrated in Figure 3.7, the time of interest is the time difference when the GA server receives the hot key and when the first corresponding frame is rendered in the GA server. Combined with Figure 3.1, we can see that, in the experiments, a physical machine runs VMware Horizon View Client to launch GPU virtualization instances, while both tested games and the GamingAnywhere server runs on these instances. When measuring the response delay of a tested game on a tested GPU virtualization solution, the GA server receives the command of displaying the main menu from the GA client and lets the tested game handle this command. Then the GA server renders new generated frames on the GPU virtualization instance, as well as sends the results back to the GA client. In the experiments, however, the only time of interest is the time difference between when the GA server receives the command of showing the main menu, and the time when the tested game renders the first frame that corresponds to this command, which equals to the time difference (*t2* - *t1*) in Figure 3.6. Therefore, the processing delay of the server (*t2* - *t1*) in Figure 3.6 is considered to be the response delay in the experiments.

To determine the response delay, a function call to capture the system time is inserted into two places at the server side. One place is where the key event is generated, the other one is the place where the frames are rendered. The former one corresponds to *t1*, while the latter one corresponds to *t2*. Instead of capturing the event of displaying the main menu, other key events in the tested games are utilized to measure response delay, the shooting event, jumping event and moving event are used to evaluate the response of delay

AssaultCube, LEGO Batmen 2 and Warhammer 40,000 respectively. For instance, the Figure 3.8 shows the shooting event in AssaultCube. The time difference between when the shooting command is created (the character holds the gun and stays still at this time as shown in Figure 3.8(a)) and when the character starts shooting (as shown in Figure 3.8(b)) is considered the response delay of AssaultCube. In the experiments, the response delay samples for these games are collected manually as the first frame that corresponds to the tested key events cannot be found automatically.

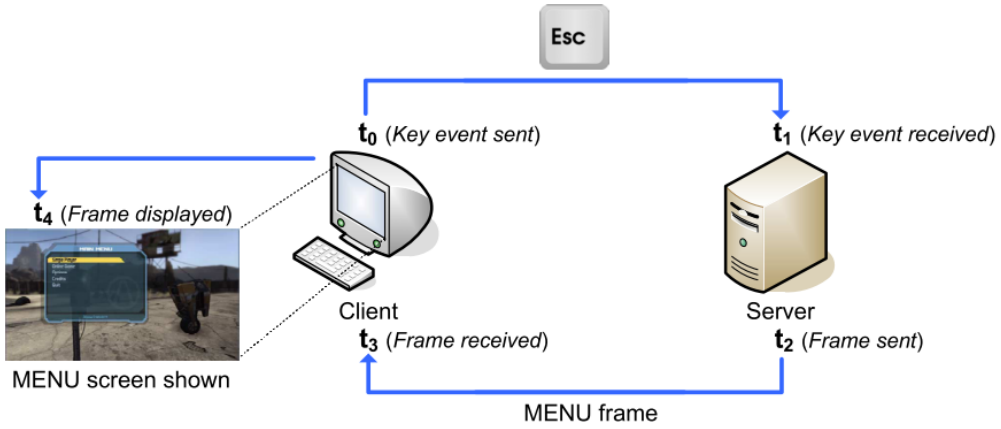


Figure 3.6: The Main Events in the Measurement of the Response Delay of a Cloud Gaming Platform by Invoking the Menu Screen [7]

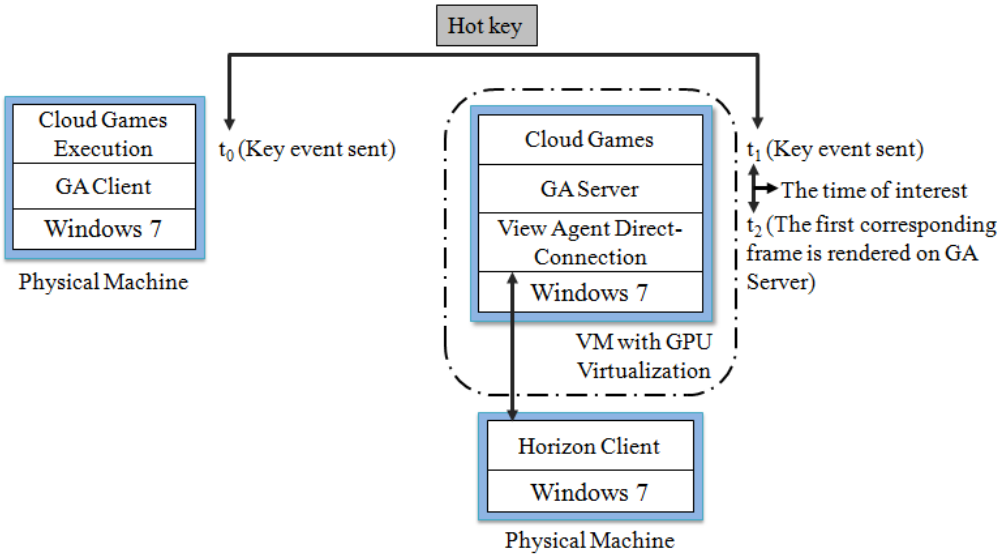
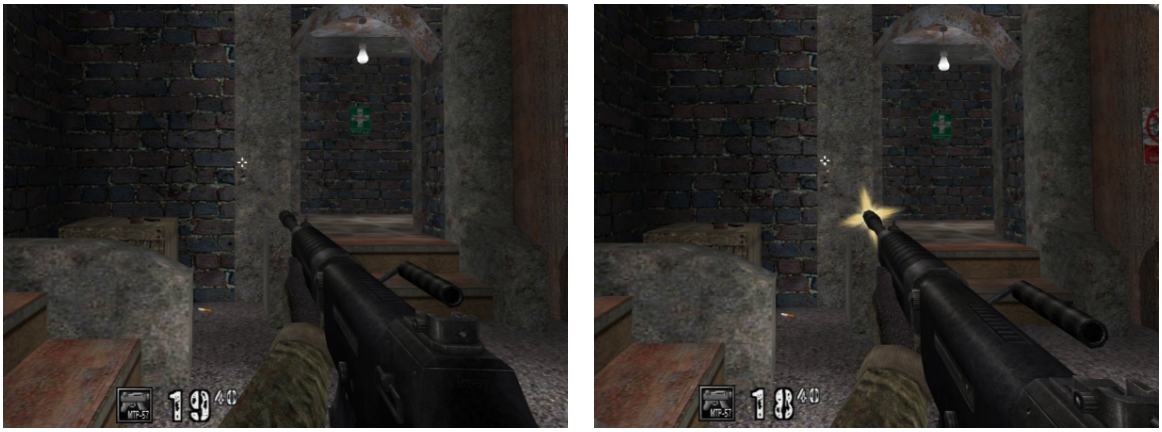


Figure 3.7: The Time of Interest in the Experiments

3.4.4 Measuring Image Quality

A simple metric, the Peak-Signal-to-Noise-Ratio (PSNR) [45], is adopted to measure image quality. The basic idea is to compute the PSNR metric for each of a number of frames captured at the client side, using



(a) Before Shooting (b) Start Shooting

Figure 3.8: The Triggering Event Used to Record Response Delay in AssaultCube

the difference between it and the corresponding frame captured from the original game video at the server side. In other words, the PSNR method quantifies the amount of error (noise) in the reconstructed video. The PSNR is derived using the mean squared error (MSE) in relation to the maximum possible value of the luminance (here the value is 255) as shown in equation (3.1) and equation (3.2). Here $f_{i,j}$ is the original signal at pixel (i, j), $F_{i,j}$ is the reconstructed signal, and $M \times N$ is the picture size. The result is a single number in decibels.

$$MSE = \frac{\sum_{j=1}^{N} \left(\sum_{i=1}^{M} (f_{i,j} - F_{i,j})^2 \right)}{M \times N} \tag{3.1}$$

$$PSNR = 10 \log \frac{255^2}{MSE} \tag{3.2}$$

In the experiments, the pre-rendered intro movie of LEGO Batman 2 is chosen to record with the resolution 1280×800 for calculating image quality. To obtain accurate samples for image quality, a function of capturing frames is inserted in GamingAnywhere to capture a deterministic sequence of uncompressed frames from the bare metal system and GPU virtualization instance. In the experiments, no compression is performed on the captured frames for the sake of simplicity. Therefore, the original frames are captured and save as BMP file format without any compression. The PSNR method is used in the experiments to calculate the image quality. The video captured from the bare-metal system is considered the original video, while the one captured from a GPU virtualization instance is considered the reconstructed video as the GPU virtualization it adopts may cause the loss of the image quality. The PSNR values for image quality are calculated by comparing the sequence captured from the bare-metal system and the one from a tested GPU virtualization instance. The algorithm of calculating the image quality for each GPU virtualization solution is described as follows.

1. Play the pre-rendered introduction movie of LEGO Batman 2 on the bare-metal system and the tested GPU virtualization instance (for example the vDGA instance) respectively.
2. Capture the generated frames during the play of the intro movie.
3. Automatically find the same frame from the bare-metal system and the tested GPU virtualization instance, then obtain a PSNR value from these two frame.
4. Obtain 100 PSNR values by repeating the step 3 to process a deterministic sequence of uncompressed frames that is captured from the bare-metal system and the tested GPU virtualization instance.
5. Measure the image quality for the tested GPU virtualization instance from these 100 PSNR values.

Chapter 4

Experimental Results and Discussion

All the experimental results are discussed in this chapter. Section 4.1 provides the baseline configuration resource usage. Section 4.2 describes the results of each single GPU virtualization instance, and Section 4.3 discusses the results of double GPU virtualization instances for the test of potential scalability. The graphics card benchmarks are run fifteen times in each tested configuration, and the average and standard deviation are calculated. As for cloud games, 50 response delay samples are collected for each tested game in each tested configuration. Moreover, to calculate image quality, 100 frame samples are captured from the pre-rendered intro movie of LEGO Batman in each tested configuration.

4.1 Baseline

As mentioned in the previous chapter, the pure hardware resource consumption of each graphics card benchmark is calculated using its total consumption minus the baseline. It is not reasonable to compare the hardware consumption of bare-metal system with that of GPU virtualization instances as they are configured with different hardware. Therefore, the hardware resource consumption is analyzed in two aspects. Firstly, the hardware consumption of the physical machine (Machine 2) is recorded when it runs the same graphics card benchmark locally (bare-metal system) and remotely on GPU virtualization instances via VMware Horizon View (GPU virtualization instances). These results are compared to find the pattern of hardware consumption of the physical machine for running each tested configuration. Secondly, the hardware consumption of each GPU virtualization instance is also recorded. These results are compared and analyzed to find the main bottleneck in each tested GPU virtualization instance. The expected pattern of hardware consumption of each GPU virtualization instance is that the more hardware resource an instance consumes, the better performance it should achieve.

Table 4.1 shows the baseline hardware resource consumption of each tested configuration. This baseline data is calculated from three one-minute records of hardware consumption of each tested configuration (3×60 samples in total) when running no benchmark. This data is used to calculate pure hardware consumption of each tested configuration. To ensure pure hardware consumption of each benchmark in each tested configuration is objectively measured, the samples of baseline and the total hardware consumption of each benchmark in each tested configuration are obtained in the same run.

From the results, we can see that the hardware consumption in each tested configuration looks stable except for the GPU usage of the bare-metal system and the CPU usage of the vSGA instance, where the standard deviation is larger than the mean. This is because, in the bare-metal system, the GPU remains almost idle all the time as no benchmark or tested game is running during the period of measuring the baseline of the hardware consumption. Nevertheless, there are few samples which GPU is not idle, causing the standard deviation higher than the mean. Similarly, the CPU usage of the vSGA instance remains between 2% and 3% for the most of the time, but there are some samples which CPU usage is much higher the average, causing the same problem.

Table 4.1: Base Hardware Resource Consumption of each Tested Configuration

Tested Configuration		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
GPU (%)	Mean	0.1	1.3	9.7	9.7	9.6	11.2
	Std. Dev.	0.34	0.88	0.48	0.47	0.48	0.90
Graphics Memory (MB)	Mean	99.1	199.9	156.3	188.3	380.3	143.3
	Std. Dev.	0.03	0.35	0.00	0.00	0.00	0.00
CPU (%)	Mean	26.8	2.3	3.5	2.7	3.1	3.8
	Std. Dev.	8.53	7.51	3.43	3.39	3.34	3.46
RAM (MB)	Mean	1,405.3	1035.1	852.0	919.6	866.9	1,002.5
	Std. Dev.	17.47	10.09	11.80	8.34	14.62	8.68

4.2 The Results of the First Group of Experiments

4.2.1 The Results of Doom 3 Benchmark

Table 4.2 shows the frames per second results for the Doom 3 benchmark. We can see that the vSGA instance performs extremely poor at this benchmark. It takes 778.7 seconds on average to run this test and only performs at 2.9 frames per second. Secondly, all vGPU instances obtain good performance in this benchmark. In particular, the more GPU resources a vGPU instance has, the better result it obtains. Additionally, vGPU K180Q instance behaves a little better than the bare-metal system. Thirdly, the vDGA instance gains the best performance among all tested configurations. Fourthly, the stability of all tested configurations is good as the standard deviation of any results is less than 3 frames per second, and in all cases, less than 5% of the mean frame rate.

Table 4.3 shows the hardware consumption of Machine 2 when it runs the Doom 3 benchmark locally and remotely on GPU virtualization instances via VMware Horizon View. When run locally, the local GPU resource usage is (not surprisingly) much higher than when run remotely. When using vSGA, the local GPU

usage is only 1.3%, and it is less than 10% in the vDGA all vGPU instances. Moreover, all configurations using remote GPU virtualization instances consume less than 13 MB of graphics memory. This is because the main graphics calculations occur on GPU virtualization instances, and the physical machine only utilizes GPU resources to render updated frames it receives from those GPU virtualization instances. Secondly, the vDGA and all vGPU instances utilize more CPU and RAM resources than the bare-metal system. This may be because the physical machine needs those extra CPU and RAM resources to decode and render the received frames. Thirdly, the vSGA instance consumes very little CPU but a little more RAM resources compared to the bare-metal system. This may be because the vSGA instance handles the Doom 3 benchmark so slowly that physical machine does not need to use too many CPU resources to deal with updated frames.

Table 4.2: Doom 3 Benchmark

Tested Configuration		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
Frames Per Second (fps)	Mean	55.4	2.9	50.2	51.4	56.4	58.9
	Std. Dev.	0.13	0.04	1.06	0.9	2.06	0.71

Table 4.3: Machine 2 H/W Resource Consumption: Doom 3 Benchmark

Tested Configuration		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
GPU (%)	Mean	93.9	1.3	9.8	9.9	9.8	9.6
	Std. Dev.	17.83	1.45	0.83	0.45	0.53	1.35
Graphics Memory (MB)	Mean	243.4	10.7	12.4	11.6	11.6	11.6
	Std. Dev.	0.97	0.04	0.00	0.03	0.03	0.00
CPU (%)	Mean	122.2	17.8	187.0	190.7	194.9	177.0
	Std. Dev.	28.72	29.91	43.63	35.81	36.35	45.45
RAM (MB)	Mean	292.2	367.1	333.3	338.9	345.7	337.5
	Std. Dev.	22.47	3.86	3.36	3.28	2.93	3.57

Pure hardware resource consumption of the Doom 3 benchmark in each GPU virtualization instance is shown in Table 4.4. From the table, we can see that the vSGA instance uses 93.7% of the GPU resources to handle Doom 3 benchmark but has extremely poor performance. Moreover, it consumes fewer graphics memory, CPU and RAM resources than other GPU virtualization instances. This is because the vSGA's GPU is the bottleneck for running the Doom 3 benchmark, so that it does not need too many other hardware resources to handle it. All in all, the result shows vSGA is unable to handle Doom 3 benchmark well.

The graphics memory usage, CPU usage and RAM usage of three vGPU instances are very similar. Nevertheless, their GPU usages are different. Although both of them consume about 50% of the GPU

resources to run the Doom 3 benchmark, the vGPU K180Q instance actually uses more GPU resources than the vGPU K140Q and vGPU K120Q instances as it is equipped with a GPU that has more CUDA cores. Similarly, the vGPU K140Q instance uses more GPU resources than the vGPU K120Q instance. Combined with Table 3.5, we can see that the vGPU K120Q, vGPU K140Q and vGPU K180Q instances utilize about 12, 25 and 98 CUDA cores to run the Doom 3 benchmark respectively. The more GPU resources an instance consumes, the better performance it achieves. This explains why the vGPU K180Q instance performs better than the vGPU K140Q and the vGPU K120Q instances, and the vGPU K140Q instance behaves better than the vGPU K120Q instance.

The vDGA instance consumes more GPU resources than the vGPU K180Q instance. This may be because, unlike the vGPU K180Q instance where the graphics card driver still needs to communicate with the physical GPU via a vGPU manager, the driver on the vDGA instance directly talks to the physical GPU. Therefore, the vDGA instance is able to utilize the physical GPU in a more efficient way to produce better result than the vGPU K180Q instance even though they have the same GPU resources. Additionally, the vDGA instance consumes more CPU resources than the vGPU K180Q instance to help itself achieve better performance. Moreover, the vDGA instance uses fewer graphics memory resources. This may be because the Doom 3 benchmark in the vDGA instance is so fast that few graphics memory resources are needed.

Table 4.4: Hardware Resource Consumption of Doom 3 Benchmark

Tested Configuration		**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
GPU (%)	Mean	93.7	48.6	51.2	51.0	56.8
	Std. Dev.	12.96	19.64	14.59	13.21	14.96
Graphics Memory (MB)	Mean	204.1	231.3	231.3	231.3	115.2
	Std. Dev.	2.58	0.00	0.00	0.00	5.49
CPU (%)	Mean	30.0	102.4	104.2	111.3	173.8
	Std. Dev.	16.02	13.50	16.00	13.46	19.44
RAM (MB)	Mean	103.4	275.6	254.5	256.2	282.7
	Std. Dev.	31.43	10.76	37.05	36.39	11.56

4.2.2 PassMark PerformanceTest

The Results of PassMark PerformanceTest 2D Benchmark

Table 4.5 shows the results of the PassMark PerformanceTest 2D benchmark. From the results, we can see that the bare-metal system behaves poorly compared to other tested configurations. It achieves a much lower score than any GPU virtualization instance in all 2D benchmarks. Additionally, the vSGA instance performs a little worse than the vDGA and all vGPU instances in Simple Vectors, Complex Vector, Image Filter,

Image Rendering, and DirectX 2D, but gains the highest score in Fonts and Text and Window Interface. It is possible this is because vSGA is mainly designed for Knowledge Workers, and used to handle less graphics-intensive applications such as Microsoft Office, Web Browsers, etc. Therefore, it is not surprising that the vSGA instance is good at handling text processing, even though it has fewer GPU resources than other GPU virtualization instances. Moreover, the vDGA and all vGPU instances perform better than the bare-metal system, but the results they achieve are very similar, which means the vGPU K140Q, vGPU K180Q and vDGA instances do not fully leverage GPU resources they have to gain better results. Furthermore, each tested configuration exhibits excellent stability while running this benchmark. Except for the results of 2D Graphics Mark, Fonts and Text and Direct 2D in the vSGA instance, where the standard deviation values are 4.1%, 6.0% and 20.1% of the mean values, respectively, the standard deviation values are less than 3.0% of the corresponding mean values.

Table 4.5: PassMark PerformanceTest 2D Benchmark (Score)

Tested Configuration		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
2D Graphics Mark	Mean	167.3	555.1	641.1	648.4	665.5	642.1
	Std. Dev.	0.95	22.97	3.24	3.91	4.72	5.47
Simple Vectors	Mean	6.4	30.1	33.9	33.0	34.1	33.6
	Std. Dev.	0.06	0.54	0.71	0.61	0.72	0.67
Complex Vectors	Mean	32.3	97.2	142.4	138.2	141.4	131.9
	Std. Dev.	0.36	1.67	1.13	1.24	1.57	1.82
Fonts and Text	Mean	49.4	174.6	157.3	153.9	158.3	161.0
	Std. Dev.	0.55	10.58	1.28	2.37	1.09	1.56
Window Interface	Mean	29.1	103.6	83.7	81.2	83.5	82.6
	Std. Dev.	0.39	0.94	1.26	1.17	1.09	1.48
Image Filter	Mean	164.6	683.5	691.2	688.5	691.1	697.8
	Std. Dev.	1.75	21.99	3.03	3.96	2.64	2.57
Image Rendering	Mean	163.6	610.9	622.1	618.5	629.9	625.9
	Std. Dev.	0.84	9.75	3.38	3.31	4.19	8.08
Direct 2D	Mean	4.9	11.7	23.6	23.6	24.3	20.2
	Std. Dev.	0.01	2.25	0.37	0.43	0.38	0.27

The Results of PassMark PerformanceTest 3D Benchmarks

Table 4.6 and Table 4.7 show the results of the PassMark PerformanceTest 3D Simple and Complex Benchmark respectively. From the tables, we can see that the performance of the vSGA instance is close to that of the bare-metal system in the simple version of benchmark, and it only loses a little performance compared

to the bare-metal system in the complex version of benchmark. That is because, in the complex version of benchmark, the vSGA instance takes a little more time to generate frames and suffers more jitter[1] than the bare-metal system.

All vGPU instances provide better results than the bare-metal system and the vSGA instance. Additionally, like the results of the 2D benchmark, all vGPU instances achieve very close results in both simple and complex versions of the benchmarks, in terms of frame rate, frame render time, jitter and jitter percentage. This means the vGPU K140Q instance and the vGPU K180Q instance fail to fully utilize the additional GPU resources they have.

As well, the vDGA instance gains the best performance among all the tested configurations in both simple and complex version of benchmark. In the simple version, the vDGA instance uses less time to generate frames than other GPU virtualization instances. In addition, there is less jitter in the result of vDGA compared to that of other instances. These make the vDGA instance achieve better performance than other instances in simple version. In the complex version, although the vDGA instance uses almost the same time to generate the benchmark's frames as that of all vGPU instances, there are lower jitter values in the vDGA's result, which makes it achieve a little better performance than vGPU instances. Like the results of the Doom 3 benchmark, all GPU virtualization instances show good stability as they all achieve predicatable frame rate with low standard deviation less than 3% of the mean frame rate in all cases in both simple and complex versions of the PassMark PerformanceTest 3D benchmarks.

Table 4.6: PassMark PerformanceTest 3D Simple Benchmark

Tested Configurations		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
Frames Per Second (fps)	Mean	30.6	29.8	41.7	44.0	41.8	51
	Std. Dev.	0.05	0.2	0.68	0.88	0.56	1.07
Frame Render Time (ms)	Mean	32.7	33.5	24.0	22.8	23.8	19.6
	Std. Dev.	0.05	0.23	0.4	0.45	0.32	0.44
Jitter (ms)	Mean	1.5	1.1	3.6	2.8	3.1	1.8
	Std. Dev.	0.04	0.37	0.81	0.47	0.44	0.26
Jitter Percentage	Mean	4.7	3.3	14.0	12.4	13.0	9.0
	Std. Dev.	0.14	1.09	3.5	2.17	1.81	1.12

Table 4.8 and Table 4.9 respectively show the hardware consumption of Machine 2 for running the PassMark PerformanceTest 3D simple and complex benchmarks. Like the hardware consumption of Doom 3, all GPU virtualization instances consume fewer GPU resources and less graphics memory but more CPU resources than the bare-metal system in both simple and complex benchmarks for the same reason: all the

[1]The jitter jitter in this context is the absolute deviation from the mean frame render time.

Table 4.7: PassMark PerformanceTest 3D Complex Benchmark

Tested Configuration		Bare-metal	vSGA	vGPU K120Q	vGPU K140Q	vGPU K180Q	vDGA
Frames Per Second (fps)	Mean	15.4	14.5	35.9	35.8	35.7	36.4
	Std. Dev.	0.03	0.19	0.16	0.28	0.2	0.18
Frame Render Time (ms)	Mean	64.8	69.2	27.9	27.9	28.0	27.4
	Std. Dev.	0.15	0.89	0.12	0.22	0.16	0.13
Jitter (ms)	Mean	0.5	3.8	3.1	3.2	3.1	2.5
	Std. Dev.	0.11	0.19	0.19	0.14	0.2	0.09
Jitter Percentage	Mean	0.8	5.5	11.1	11.6	11.1	9.1
	Std. Dev.	0.16	0.23	0.71	0.52	0.72	0.29

graphics computing occurs on GPU virtualization instances instead of on the physical machine. Additionally, the hardware consumption of Machine 2 varies according to the GPU virtualization it launches. For example, the vSGA instance consumes more CPU but slightly less RAM than other GPU virtualization instances in these two benchmarks. Another example is that the vDGA instance utilizes fewer CPU resources than other GPU virtualization instances.

Table 4.8: Machine 2 H/W Resource Consumption: Performance 3D Simple Benchmark

Tested Configuration		Bare-metal	vSGA	vGPU K120Q	vGPU K140Q	vGPU K180Q	vDGA
GPU (%)	Mean	71.5	7.2	6.8	7.5	7.4	7.2
	Std. Dev.	10.07	3.19	0.82	1.57	0.88	1.23
Graphics Memory (MB)	Mean	106.1	10.8	12.4	11.6	11.6	11.7
	Std. Dev.	2.92	0.03	0.03	0.03	0.00	0.00
CPU (%)	Mean	27.0	172.4	162.7	154.5	154.3	103.7
	Std. Dev.	12.11	102.76	36.19	35.61	34.03	36.45
RAM (MB)	Mean	299.2	197.2	204.6	217.2	206.1	207.4
	Std. Dev.	3.19	23.23	30.99	22.16	19.49	22.20

Table 4.10 contains the hardware resource consumption of the PassMark PerformanceTest 3D simple benchmark in each GPU virtualization instance. Compared to other GPU virtualization instances, the vSGA instance consumes fewer GPU but more CPU resources. This may be due to the fact that, unlike the Doom 3 benchmark, which simply renders a preset number of frames as fast as it can, the PerformanceTest 3D benchmark also processes some game logic such as in-game physics. In this case, the vSGA instance may not be good at dealing with benchmark's logic and thus consumes more CPU. Moreover, with this restriction,

Table 4.9: Machine 2 H/W Resource Consumption: Performance 3D Complex Benchmark

Tested Configuration		Bare-metal	vSGA	vGPU K120Q	vGPU K140Q	vGPU K180Q	vDGA
GPU (%)	Mean	55.9	7.2	6.9	6.0	7.0	7.0
	Std. Dev.	2.66	3.18	0.89	0.80	0.70	0.58
Graphics Memory (MB)	Mean	118.4	10.8	12.4	11.7	11.6	11.6
	Std. Dev.	0.03	0.03	0.03	0.02	0.03	0.03
CPU (%)	Mean	53.9	173.3	131.4	132.5	132.7	112.0
	Std. Dev.	10.98	99.72	32.06	28.30	26.26	27.85
RAM (MB)	Mean	267.8	198.4	326.8	329.6	304.7	304.1
	Std. Dev.	20.00	21.82	18.53	5.28	24.87	23.99

its GPU cannot be fully used as it may have to wait until the CPU finishes handling the game logic. This can also explain why the vSGA instance performs worse than other GPU virtualization instances.

All vGPU instances use very similar hardware resources in terms of graphics memory, CPU and RAM. Nevertheless, the vGPU K140Q instance and the K180Q instance utilize more GPU resources but do not gain better results than vGPU K120Q instance. Additionally, none of GPUs in vGPU cases is fully loaded. This may be because the GPUs in vGPU instances are not the main bottleneck that stops the achievement of higher performance in this benchmark.

The vDGA instance utilizes more GPU and more RAM resources to run the benchmark. More GPU resources help it to process graphics computation much faster, while more RAM resources help it handle the game logic much faster. Therefore, the vDGA instance gains the best performance among all tested configurations in simple version of the PassMark PerformanceTest 3D benchmark.

Table 4.11 describes the hardware resource consumption of the PassMark PerformanceTest 3D complex benchmark in each GPU virtualization instance. The vSGA instance uses almost an entire GPU to run the benchmark but achieves poor results compared to other GPU virtualization instances, which means that the GPU in the vSGA instance may be the primary bottleneck to run this benchmark.

Like the vSGA instance, all vGPU instances also use the entire GPU to run the benchmark. Nevertheless, they achieve very similar results, which means the instance that has more GPU resources does not perform better the one that has fewer GPU resources. Additionally, as all vGPU instances are equipped with the same CPU and their CPU usages are similar, these vGPU instances may handle the benchmark's logic at the same speed, which may be the primary bottleneck that stops the vGPU instances that have more GPU resources from achieving better results. However, as the assumption above is not proved, further analysis is required to find the real cause for the difference.

Like all vGPU instances, the vDGA instance also uses almost the entire GPU to run the benchmark but obtains a similar frame rate compared with all vGPU instances, which means the vDGA's GPU is not the

bottleneck that prevents the vDGA instance from achieving better results (higher frame rate). In addition, the vDGA instance uses more CPU and RAM resources, which may be the reason why the vDGA instance has lower jitter values than other GPU virtualization instances.

Table 4.10: Hardware Resource Consumption of PerformanceTest 3D Simple Benchmark

Tested Configuration		**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
GPU (%)	Mean	63.6	71.9	71.4	71.3	86.3
	Std. Dev.	9.34	13.82	12.66	15.52	5.06
Graphics Memory (MB)	Mean	65.2	71.8	71.1	68.8	71.1
	Std. Dev.	0.31	0.00	0.00	1.87	0.00
CPU (%)	Mean	94.6	64.3	62.5	64.1	65.9
	Std. Dev.	40.65	6.97	10.95	6.27	13.97
RAM (MB)	Mean	136.3	125.8	110.1	118.6	195.9
	Std. Dev.	11.97	2.56	4.13	3.70	10.40

Table 4.11: Hardware Resource Consumption of Performance 3D Complex Benchmark

Tested Configuration		**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
GPU (%)	Mean	93.2	88.6	88.4	89.1	87.8
	Std. Dev.	5.22	5.10	4.47	3.50	0.11
Graphics Memory (MB)	Mean	89.3	86.3	85.5	83.4	85.5
	Std. Dev.	0.06	0.00	0.03	0.01	0.00
CPU (%)	Mean	67.5	82.7	78.5	80.9	102.2
	Std. Dev.	14.23	9.71	10.08	9.22	23.27
RAM (MB)	Mean	121.9	150.5	130.8	145.9	231.5
	Std. Dev.	7.05	4.11	3.63	3.11	40.20

4.2.3 Unigine Sanctuary Benchmark

Table 4.12 shows the results of the Unigine Sanctuary benchmark tests. The bare-metal system performs worse than all GPU virtualization instances in these configurations. The vSGA instance cannot run the benchmark under the middle and high configurations as it does not support the features of multiple-sampling and Anti-aliasing.[2] Additionally, it performs at 29.3 frames per second in the low configuration, which is a little better than the bare-metal system but about 10 fps fewer than the vDGA and all vGPU instances.

[2]These two features are used in the middle and high configurations of the Unigine Sanctuary benchmark.

Except for the vGPU K120Q instance that performs a little worse than than the vDGA and other vGPU instances in the low configuration, the performance of the vDGA and all vGPU instances are very close to each other in these three configurations. This means the vGPU K140Q, vGPU K180Q and vDGA instances fails to leverage the advantage of having more GPU resources to obtain better results.

Table 4.13 shows the hardware consumption of Machine 2 for running Unigine Sanctuary benchmark in the low configuration. Like the hardware consumption of the physical machine for running Doom 3 and PassMark 3D benchmarks, all GPU virtualization instances consume less GPU and graphics memory but more CPU resources than the bare-metal system. It also shows that the hardware consumption of Machine 2 varies according to the GPU virtualization instance it launches. For instance, in this case, the vSGA instance consumes more CPU and RAM than other GPU virtualization instances. The implementation of these vGPU techniques may be the reason that causes these differences. For instance, combined with the hardware consumption of Machine 2 when launching vGPU virtualization instances to run the Doom 3, PassMark PerformanceTest 3D and Unigine benchmarks, the vDGA instance always consumes fewer CPU resources than other GPU virtualization instances.

Like the results of the PassMark PerformanceTest 3D benchmarks, all GPU virtualization instances also show the good stability in the Unigine benchmark under all configurations. The standard deviation of the results of the bare-metal system and vSGA instance is between 2% and 5% of the mean frame rate, while it is less than 2% of the mean frame rate in vDGA and all vGPU instances.

Table 4.12: Unigine Sanctuary Benchmark (Frames/Second)

Tested Configuration		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
Low Configuration	Mean	27.9	29.3	36.1	40.7	40.5	40.6
	Std. Dev.	0.91	1.15	0.06	0.06	0.07	0.18
Medium Configuration	Mean	20.3	NA	31.2	31.3	31.1	31.3
	Std. Dev.	0.34	NA	0.04	0.09	0.04	0.06
High Configuration	Mean	15.3	NA	24.5	24.6	24.6	24.6
	Std. Dev.	0.63	NA	0.31	0.0	0.04	0.05

Tables 4.14, 4.15 and 4.16 show the hardware resource consumption of Unigine Sanctuary benchmark in low, middle and high configuration respectively in each GPU virtualization instance. The vSGA instance consumes 90.8% GPU and more CPU resources than other GPU virtualization instances in low configuration, but does not gain the same good result as others. Nevertheless, this may be the best performance that vSGA can produce, which further proves that the vSGA instance is not capable of running 3D applications.

The pattern of hardware resource consumption in the vGPU and vDGA instances is very similar under those configurations. The GPU load in these instances is close to 100% and they utilize similar graphics memory, CPU and RAM. Nevertheless, like the results of the PassMark PerformanceTest 3D complex bench-

Table 4.13: Machine 2 H/W Resource Consumption: Unigine Sanctuary Benchmark in Low Configuration

Tested Configuration		Bare-metal	vSGA	vGPU K120Q	vGPU K140Q	vGPU K180Q	vDGA
GPU (%)	Mean	91.8	8.7	6.1	5.7	8.8	8.0
	Std. Dev.	3.40	0.80	3.74	1.31	1.06	1.17
Graphics Memory (MB)	Mean	189.8	10.8	12.4	11.7	11.6	7.5
	Std. Dev.	0.23	0.03	0.05	0.02	0.03	0.03
CPU (%)	Mean	53.5	210.5	166.0	173.4	180.1	142.6
	Std. Dev.	12.76	65.40	43.07	44.42	51.21	36.36
RAM (MB)	Mean	165.9	199.6	154.5	168.9	158.4	163.2
	Std. Dev.	13.05	9.61	4.56	5.77	5.55	33.09

marks, the instances that have more GPU cores do not achieve better performance. One potential reason is that, these instances handle the benchmark's logic at the same speed as the CPUs in these instances are the same and they consume the same CPU resources. This may be the reason that these instances obtain similar results in this benchmark under all configurations. Nevertheless, as this is just an assumption, more detailed knowledge of hardware resource consumption is required to figure out the reason that causes this problem.

Table 4.14: Hardware Resource Consumption of Unigine Sanctuary Benchmark: Low Configuration

Tested Configuration		vSGA	vGPU K120Q	vGPU K140Q	vGPU K180Q	vDGA
GPU (%)	Mean	90.8	90.3	90.2	90.2	87.8
	Std. Dev.	5.91	0.28	0.36	0.34	0.00
Graphics Memory (MB)	Mean	158.7	154.0	154.3	154.4	153.5
	Std. Dev.	13.34	14.78	13.73	13.49	3.18
CPU (%)	Mean	85.8	64.6	63.9	63.7	63.1
	Std. Dev.	10.42	10.27	9.91	9.79	8.31
RAM (MB)	Mean	142.7	79.1	88.2	85.8	80.9
	Std. Dev.	4.17	11.73	10.59	7.76	11.69

4.2.4 Extra Experiments

From the results of the PassMark PerformanceTest 3D complex benchmark and Unigine Sanctuary benchmark, we can see that the vDGA and all vGPU instances produce similar results. There must be some bottleneck. One possible reason is the advanced features of graphics card which are enabled in the experi-

Table 4.15: Hardware Resource Consumption of Unigine Sanctuary Benchmark: Middle Configuration

Tested Configuration		**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
GPU (%)	Mean	90.1	90.1	90.2	87.7
	Std. Dev.	1.43	1.21	0.37	1.55
Graphics Memory (MB)	Mean	186.5	186.5	186.4	188.4
	Std. Dev.	16.02	16.81	15.78	3.71
CPU (%)	Mean	52.2	53.8	52.1	51.6
	Std. Dev.	9.88	10.11	10.62	8.90
RAM (MB)	Mean	97.0	109.8	94.0	88.8
	Std. Dev.	10.00	6.74	8.35	8.00

Table 4.16: Hardware Resource Consumption of Unigine Sanctuary Benchmark: High Configuration

Tested Configuration		**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
GPU (%)	Mean	90.1	89.9	89.9	87.8
	Std. Dev.	0.52	1.98	2.50	0.00
Graphics Memory (MB)	Mean	219.5	219.6	219.6	221.5
	Std. Dev.	18.05	18.33	18.33	3.69
CPU (%)	Mean	47.1	49.4	49.2	46.9
	Std. Dev.	13.80	12.65	12.59	11.89
RAM (MB)	Mean	82.1	103.2	88.1	91.8
	Std. Dev.	8.16	8.56	11.35	8.73

ments as shown in Table 3.6. These features help the benchmarks produce better image quality, but utilize more resources and degrade the frame rate performance. To explore whether these features are the main reason that prevents performance enhancement, they are disabled and the PassMark PerformanceTest 3D and Unigine Sanctuary benchmarks are run with the same configurations in the vDGA and each vGPU virtualization instance.

Table 4.17 and Table 4.18 show the results of the PassMark PerformanceTest 3D simple and complex benchmarks respectively that are re-run on the vDGA and vGPU instances without the features in Table 3.6. The vDGA and all vGPU instances obtain better results without those features. Both of them take less time to generate a frame and generate higher frame rates in both benchmarks. Additionally, the vDGA instance achieves the best performance among all tested configurations, which further shows that vDGA performs better than vGPU K180Q even through they are equipped with same GPU resources (Nvidia CUDA cores). Moreover, all vGPU instances obtain similar results in these two benchmarks in terms of frame rate, frame render time and jitter time. Therefore, those high features in Nvidia graphics cards are not the bottleneck that prevents the vGPU instance that has additional GPU resources from achieving the better performance.

Table 4.19 shows the results of the Unigine Sanctuary benchmark which are re-run in the vDGA and all vGPU instances. Like the results of the PerformanceTest 3D benchmark, the vDGA instance produces the best performance under all three configurations, which means those features in Table 3.6 are potential reasons that prevents vDGA from achieving better results. In addition, all vGPU instances also obtain similar results in this benchmark in whatever configuration, which means those features in Nvidia graphics card are not the bottleneck that limits the performance.

Table 4.17: PassMark PerformanceTest 3D Simple Benchmark

Tested Configurations		**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
Frames Per Second	Mean	57.8	58.0	58.7	60.7
	Std. Dev.	0.43	0.58	0.45	0.28
Frame Render Time (ms)	Mean	17.3	17.3	17.1	16.8
	Std. Dev.	0.13	0.17	0.13	0.08
Jitter (ms)	Mean	1.5	1.8	1.7	0.3
	Std. Dev.	0.18	0.13	0.17	0.12
Jitter Percentage	Mean	8.7	10.7	10.1	1.8
	Std. Dev.	0.99	0.64	0.89	0.69

Table 4.18: PassMark PerformanceTest 3D Complex Benchmark

Tested Configuration		**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
Frames Per Second (fps)	Mean	37.8	37.5	37.6	45.3
	Std. Dev.	0.31	0.34	0.52	0.16
Frame Render Time (ms)	Mean	26.5	26.7	26.6	22.1
	Std. Dev.	0.21	0.24	0.36	0.08
Jitter (ms)	Mean	7.5	7.2	7.1	2.2
	Std. Dev.	0.29	0.36	0.38	0.08
Jitter Percentage	Mean	28.3	26.9	26.7	10.1
	Std. Dev.	1.24	1.58	1.75	0.36

Table 4.19: Unigine Sanctuary Benchmark: Frames Per Second

Tested Configuration		**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
Low Configuration	Mean	59	59.6	59.5	65.2
	Std. Dev.	0.13	0.15	0.32	0.04
Medium Configuration	Mean	40.6	41.1	40.9	43.6
	Std. Dev.	0.32	0.23	0.32	0.12
High Configuration	Mean	30.1	30.3	30.1	31.8
	Std. Dev.	0.08	0.17	0.13	0.04

4.2.5 The Results of Cloud Games

Response Delay

Table 4.20 show the results of the response delay of the tested games in each tested configuration. The bare-metal system performs poorly at AssaultCube. The result that the bare-metal system produces is totally unacceptable as it is much higher than the threshold of FPS games (100 ms). Additionally, AssaultCube sometimes fails to response to the keyboard and mouse input when it is running on the bare-metal system, which significantly degrades the user experience. Although the bare-metal system achieves acceptable response delay in the other two games, it performs worse than all GPU virtualization instances.

The response delay of the vSGA instance (125.2 ms on average) is a little higher the threshold of FPS games, but AssaultCube in the vSGA instance always responses to the new game commands. Moreover, the vSGA instance performs at 49.3 ms and 252.3 ms respectively in LEGO Batman and Warhammer 40,000, which is better than that of the bare-metal system and much lower than the threshold of RPG games (500 ms) and RTS games (1000 ms).

All vGPU cases gain acceptable and similar results in all three games. Additionally, they perform better than the vSGA instance and much better than the bare-metal system in both three games. For instance, the average response delay of AssaultCube in all vGPU instances is below 35 ms, while the bare-metal system only performs at 381.1 ms. Moreover, the vGPU K180Q instance does not achieve the best performance among the three vGPU instances and the vGPU K140Q instance does not perform better than vGPU K120Q instance either. This may be because the GPU resource configuration for vGPU K120Q instance is over provisioned for running these three games.

The vDGA instance obtains similar results in AssaultCube and LEGO Batman that all vGPU instances achieve, but it performs better in Warhammer 40,000 than all vGPU instances. This is further evidence that vDGA technology performs better than vGPU technology when they have the same hardware resource configuration.

Table 4.20: Response Delay of the Tested Games

Tested Configuration		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
AssaultCube (ms)	Mean	381.1	125.2	32.2	28.9	27.8	30.1
	Std. Dev.	52.26	25.44	16.27	11.82	10.83	13.12
LEGO Batman (ms)	Mean	279.0	49.3	41.5	42.1	42.2	41.5
	Std. Dev.	38.66	10.93	9.63	10.67	13.03	11.37
Warhammer 40,000 (ms)	Mean	309.3	252.3	209.2	192.8	213.8	165
	Std. Dev.	90.63	57.3	47.34	42.98	58.91	27.18

For further investigating the reasons that cause the differences of the response delay of tested games

in each tested configuration, number of frames that are rendered to start handling the game commands used to measure the response delay, and the average frame render time of each tested game in each tested configuration are calculated and shown in Table 4.21 and Table 4.22 respectively.

In AssaultCube, the bare-metal system has to generate more frames to start dealing with the game command than other tested configurations, and the frame render time for this game is much higher than that of other tested configurations. This is why the bare-metal system produces higher response delay (381.1 ms on average) than other tested configurations for this game. The vSGA instance has to generate 3.3 frames on average to start handling the game command used to measure the response delay and the average frame generation time is 38.0 ms, each of which is a little more than the one in the vDGA and all vGPU instances. The vDGA and all vGPU instances obtain similar results in these two performance metrics, which make them achieve similar results in the response delay of AssaultCube.

Like the results of AssaultCube, in LEGO Batman, the bare-metal system has to render more frames and its average frame generation time is higher than other tested configurations. This is why the bare-metal system has the highest response delay (279.0 ms in average) among all tested configurations. In addition, although the vSGA instance render almost the same number of frames as the vDGA and all vGPU instances do, it uses a little more time to generate frames, which makes the vSGA instance has a little higher response delay than the vDGA and all vGPU instances. Moreover, like the results of AssaultCube, the vDGA and all vGPU instances have similar results in these performance metrics, making them obtain similar response delays in this game.

In Warhammer 40,000, although the bare-metal system generates the fewest frames to start handling the game command among all the tested configurations, the average frame generation time for this game is highest among all the tested configurations, which makes it has higher response delay than other tested configurations. Additionally, like the bare-metal system, the vSGA instance has fewer number of frames but higher frame generation time than the vDGA and all vGPU instances, which makes it has the response delay a little higher than the vDGA and all vGPU instances. Moreover, the frame generation time of the vDGA and all vGPU instances are similar, but the fact that the vDGA processes fewer frames than all vGPU instances enables it to obtain the lowest response delay among all tested configurations.

Table 4.23 shows the maximum response delay (the worst case) of the tested games in each tested configuration. From the table, we can see that the worst case of the response delay of AssaultCube in the bare-metal system is totally unaccepted. This is not surprising as the bare-metal system performs at 381.1 ms as average response delay for this game. Additionally, the maximum response delay of the other two games in the bare-metal system are lower than the thresholds respectively. In spite of this, the bare-metal system have the highest response delay for these three tested games among all the tested configurations.

The maximum response delay of AssaultCube in the vSGA instance is 197, which is almost twice the threshold of FPS games. Combined with the fact that vSGA performs at 125.2 ms as average response delay for this game, the vSGA instance cannot guarantee the acceptable response delay for AssaultCube.

Table 4.21: Number of Frames that are Rendered to Start Handling the Game Commands Used to Measure Response Delay of Tested Games

Tested Configuration		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
AssaultCube	Mean	6.0	3.3	2.1	2.2	2.0	2.1
	Std. Dev.	0.94	0.46	0.51	0.47	0.49	0.44
LEGO Batman	Mean	6.6	2.1	2.2	2.1	2.1	2.2
	Std. Dev.	0.84	0.33	0.37	0.33	0.33	0.39
Warhammer	Mean	4.6	5.3	7.5	7.1	7.4	5.7
40,000)	Std. Dev.	0.92	0.79	1.28	1.18	1.13	0.97

Table 4.22: Frame Generation Time of Tested Games

Tested Configuration		**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
AssaultCube	Mean	64.6	38.0	14.9	13.7	13.6	14.2
(ms)	Std. Dev.	9.24	5.84	5.51	5.43	4.41	3.72
LEGO Batman	Mean	42.8	23.6	19.4	20.1	19.7	19.5
(ms)	Std. Dev.	4.97	5.76	4.01	5.20	4.38	6.24
Warhammer	Mean	66.1	47.9	27.5	27.8	28.7	28.0
40,000 (ms)	Std. Dev.	9.11	13.11	4.03	4.20	5.58	3.14

In addition, the vSGA instance has 77 ms and 348 ms as the maximum response delay of LEGO Batman and Warhammer 40,000, which is lower than those of the bare-metal system. This indicates that the vSGA instance can handle these two games well.

The vDGA and all vGPU instances achieve very similar results in terms of the maximum response delay of the tested games, each of which is lower than the corresponding threshold. This indicates that the vDGA and all vGPU instances are able to process the tested games quickly even in the worst case.

Table 4.23: Maximum Response Delay of Tested Games

Tested Configuration	**Bare-metal**	**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
AssaultCube (ms)	478	197	72	73	57	73
LEGO Batman (ms)	374	78	77	77	77	78
Warhammer 40,000 (ms)	660	348	99	98	99	98

Image Quality

Table 4.24 shows the results of the image quality experiments for each GPU virtualization instance. The average image quality in all GPU virtualization instances is between 40 and 42 dB, which is good when compared to the threshold in PSNR that defines good image quality.[3] Additionally, all GPU virtualization cases obtain very similar results. This is because the same deterministic sequence of frame samples are captured to calculate image quality samples in each GPU virtualization instance, and the algorithm to calculate PSNR values always finds the frame that is the closest to the original one to produce the highest PSNR value. Actually, the bare-metal system always generate fewer frames than all GPU virtualization instances as it performs worse than all GPU virtualization instances when playing the intro movie of LEGO Batman 2. Therefore, suppose the sequence of the generated frames in the bare-metal system is "A, B, C, D", then the sequence in the tested GPU virtualization instance will be something like "A, A1, A2, B, C, D", where "A1, A2" are the frames that resembles frame "A". When handling these two sequences, frame "A" in the bare-metal system and the tested GPU virtualization instance are used to generate a PSNR value. Then the frames "A1" and "A2" in the tested GPU virtualization instance are abandoned as they resemble the frame "A". Finally, all the frames are processed in this way to obtain 100 PSNR values for the tested GPU virtualization instance. With such an approach, the algorithm used in the experiments always generates the PSNR values as high as possible.

[3]Values higher than 30 dB are considered good image quality in PSNR.

The histograms of image quality samples in each GPU virtualization instance are shown in Figure 4.1. From the results we can see the following: 1) The distribution of image quality samples in all instances are very similar. Only very few frame samples lose image quality according to the results. For instance, Figure 4.1(a) and Figure 4.1(c) show there is only one different PSNR sample between the PSNR samples of the vSGA instance and that of the vGPU K140Q instance. The vSGA instance has one more sample which is between 52 dB and 54 dB, while the vGPU K140Q instance moves it to the interval between 44 dB and 46 dB. 2) Most samples are higher than 36 dB. Although some samples in each GPU virtualization instance fall into the interval between 32 dB and 34 dB, these samples are still considered good quality according to definition of PSNR. Overall, those findings show that all GPU virtualization instances are able to render game frames with good quality.

Table 4.24: Image Quality Comparison

Tested Configuration		**vSGA**	**vGPU K120Q**	**vGPU K140Q**	**vGPU K180Q**	**vDGA**
PSNR (dB)	Mean	41.6	41.6	41.5	40.7	41.6
	Std. Dev.	5.09	5.09	4.84	3.85	5.09

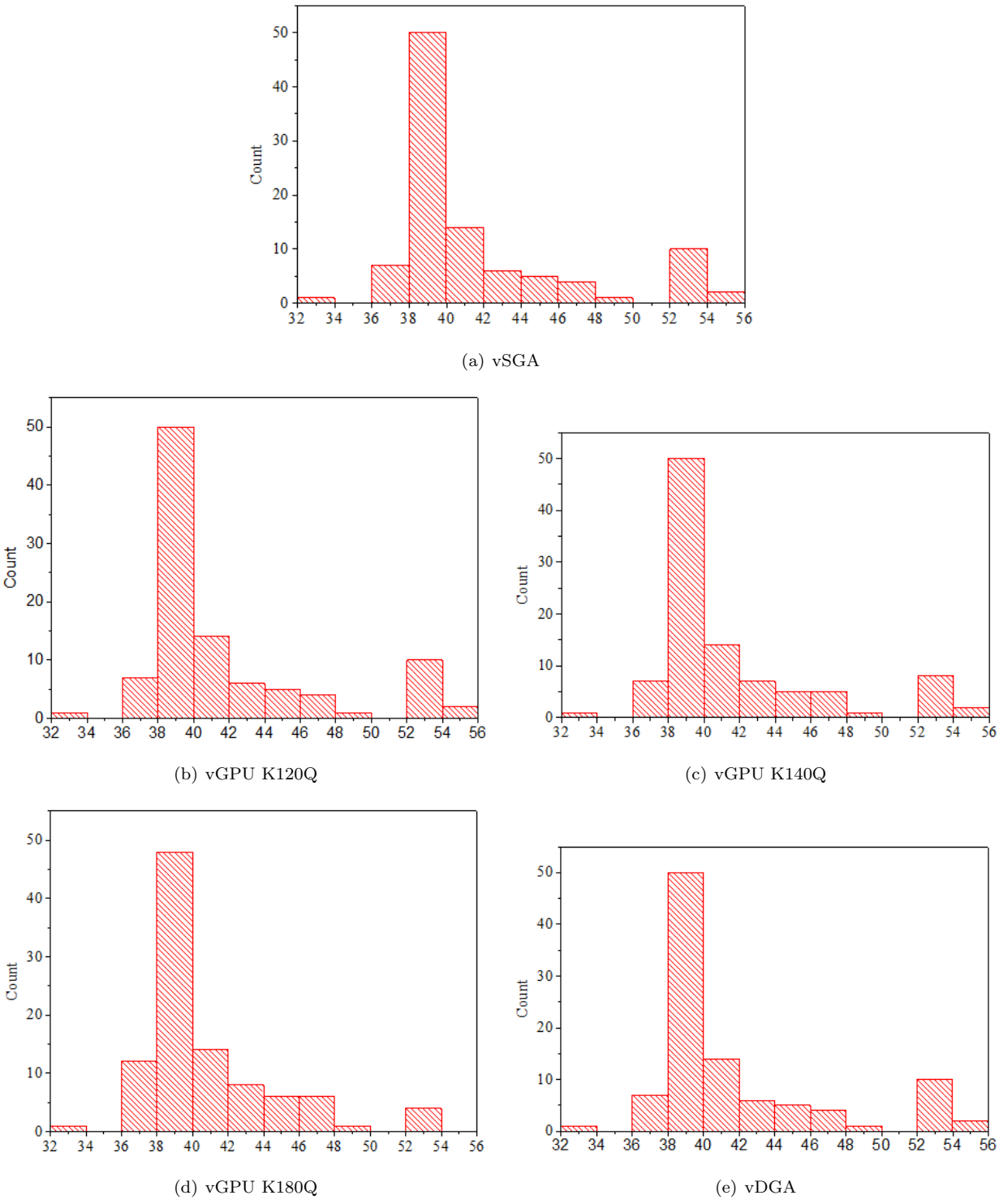
4.3 The Results of the Second Group of Experiments

4.3.1 Doom 3 Benchmark

Figure 4.2 shows the Doom 3 benchmark results in two occurrences of each tested configuration. The performance of the two bare-metal systems varies significantly, due to the hardware differences between the two physical machines. From Table 3.3 and Table 3.4, we can see that bare-metal system 1 has a faster GPU, more RAM and more hard disk space than bare-metal system 2. Therefore, it is not surprising that bare-metal system 1 performs at 55.35 fps in Doom3, while bare-metal 2 only performs at 19.77 fps on average.

All vSGA instances achieve poor performance in the Doom 3 benchmark as each vSGA instance only performs at about 3 fps on average. This indicates that vSGA's GPU is not powerful enough to handle such an benchmark.

The potential scalability of vDGA and all vGPU instances is excellent. Take the vDGA instances for example, no matter if a single instance or the first double instance running on bare-metal system 1, or the second double instance running on bare-metal system 2, all vDGA instances perform at between 58 fps and 61 fps with a standard deviation within the interval 0.71 to 2.15. Additionally, by comparing the results of bare metal system 2 and results of the second double instances of vDGA and all vGPU tested configurations, we can see that even a low-profile physical machine is able to gain high performance using vGPU or vDGA



(a) vSGA

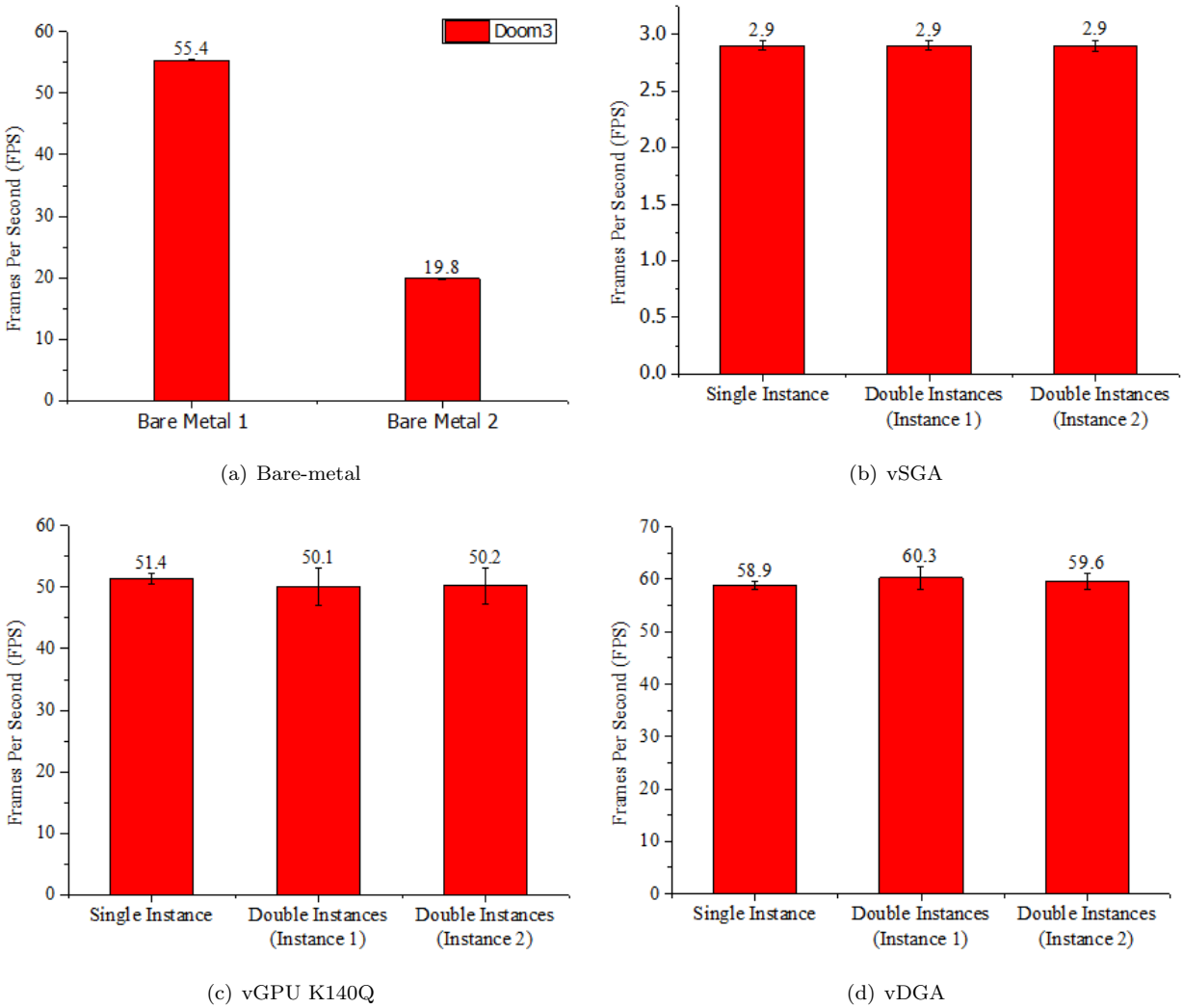
(b) vGPU K120Q

(c) vGPU K140Q

(d) vGPU K180Q

(e) vDGA

Figure 4.1: The Histograms of Image Quality



(a) Bare-metal

(b) vSGA

(c) vGPU K140Q

(d) vDGA

Figure 4.2: Doom 3 Performance

technologies.

4.3.2 Unigine Sanctuary Benchmark

Unigine Sanctuary benchmark is also tested in the second group of experiments for further analyzing the potential scalability of each GPU virtualization solution. Figure 4.3, Figure 4.4 and Figure 4.5 respectively show the results of the Unigine Sanctuary benchmark in two occurrences of vSGA, vGPU K140 and vDGA. From the results, we can see that, like the results of the Doom 3 benchmark in double instances, bare-metal system 1 performs much better than bare-metal system 2 under all configurations. Additionally, all GPU virtualization instances produce good scalability under all configurations. For instance, all vGPU K140Q instances perform between 39 fps and 41 fps in low configurations, 30 fps to 32 fps in middle configuration, and 24 fps to 25 fps in high configuration. This further shows that the potential scalability of these GPU

virtualization instances is good.

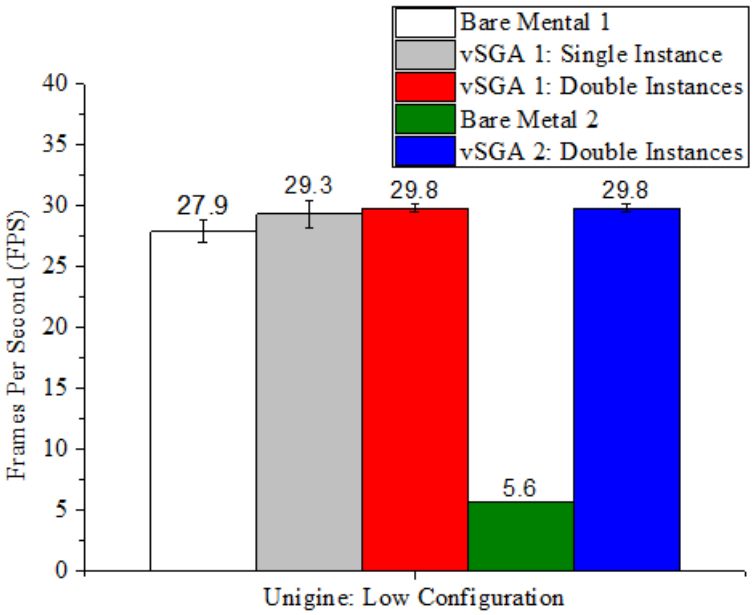


Figure 4.3: Unigine Sanctuary Performance: the vSGA Instances

4.3.3 GamingAnywhere Results

Response Delay

Only AssaultCube is used to measure response delay as here only the potential scalability of each GPU virtualization instance is the interest of the experiments. Another reason to use AssaultCube instead of the other two games is because it is a delay-intensive game, such that the performance degradation will be more obvious if these GPU virtualization instances are unstable. Figure 4.6 shows the results, from which we can see that both bare-metal systems perform poorly at AssaultCube. In particular, bare-metal 2 performs at 4938.5 ms on average and is unable to run the game smoothly. Additionally, all GPU virtualization instances perform stably in terms of AssaultCube's response delay.

Image Quality

Figure 4.7 shows the results of image quality in double instances. Like the results of response delay of AssaultCube, all GPU virtualization instances behave stably in terms of image quality. An interesting result is that the first double instance of vDGA performs a little worse than the other two vDGA instances. This may be because the frame sample generated in this instance has many samples that lose too much image quality. Nevertheless, although it performs at 37.7 dB on average, which is little lower than those of two vDGA instances, it is still considered as good image quality.

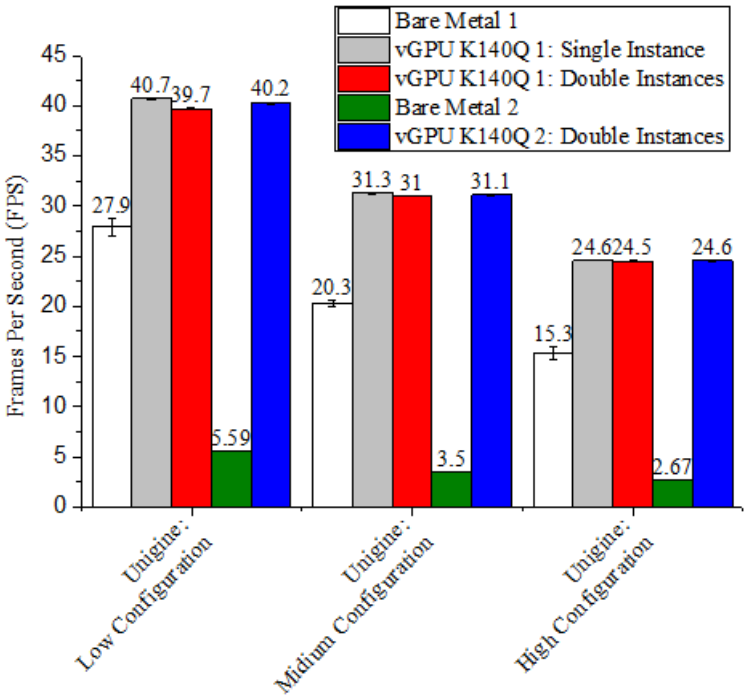


Figure 4.4: Unigine Sanctuary Performance: the vGPU K140 Instances

4.4 Summary and Analysis of Results

Based on the experiments conducted in Section 4.2 and Section 4.3, the summary and analysis of results are done and presented in this section. The followings are the main findings from the experiments.

- The experiments show the feasibility and advantage of GPU virtualization technology in cloud gaming. Firstly, it makes low-end machines able to run the games they cannot run locally. For instance, bare-metal system 2 is unable to play AssaultCube as it only produces 4938.5 ms on average as response delay for this game. With the help of GPU virtualization like vGPU K140Q, however, it can play this game smoothly with 32.7 ms on average as response delay. Secondly, while running the same application, physical machines consume fewer GPU and graphics memory resources by running it remotely on GPU virtualization instances than running it locally. For example, bare-metal system 1 consumes 93.9% of the GPU resources and 243.4 MB of graphics memory when running Doom 3 benchmark locally, but it only uses 9.9% of GPU and 11.6 MB of graphics memory when running it on vGPU K140Q instance. This turns low-end graphics cards on local machine not to be the constraint of running high-end games.
- Overall, vSGA may not be a good choice for cloud gaming. Although it achieves acceptable results in some benchmarks like the PassMark PerformanceTest 3D simple benchmark and gains good image quality and excellent response delay in LEGO Batman and Warhammer 30,000, its terrible results at the Doom 3 benchmark and Performance 3D complex benchmark and the fact that it does not support certain GPU features, like Anti-aliasing, significantly limit its performance in cloud gaming. Moreover, the fact that

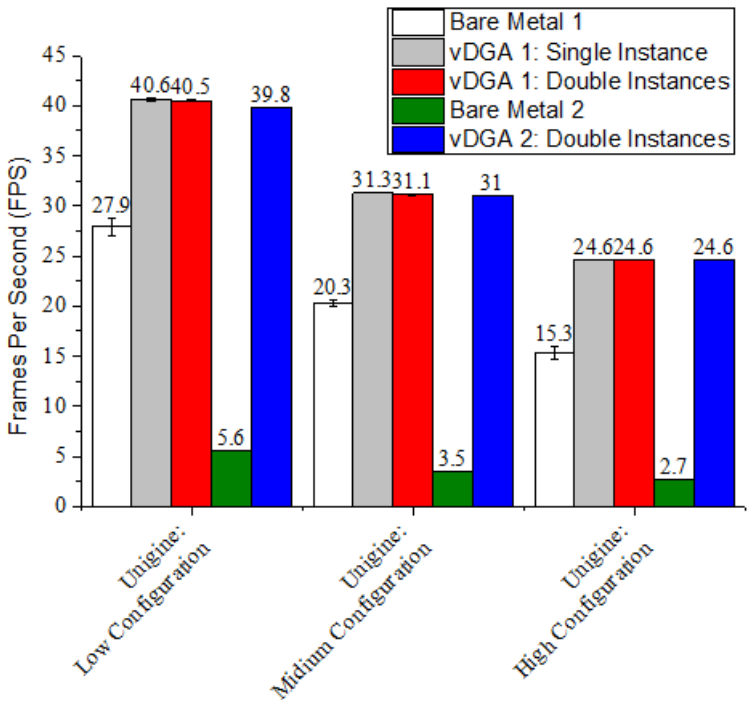
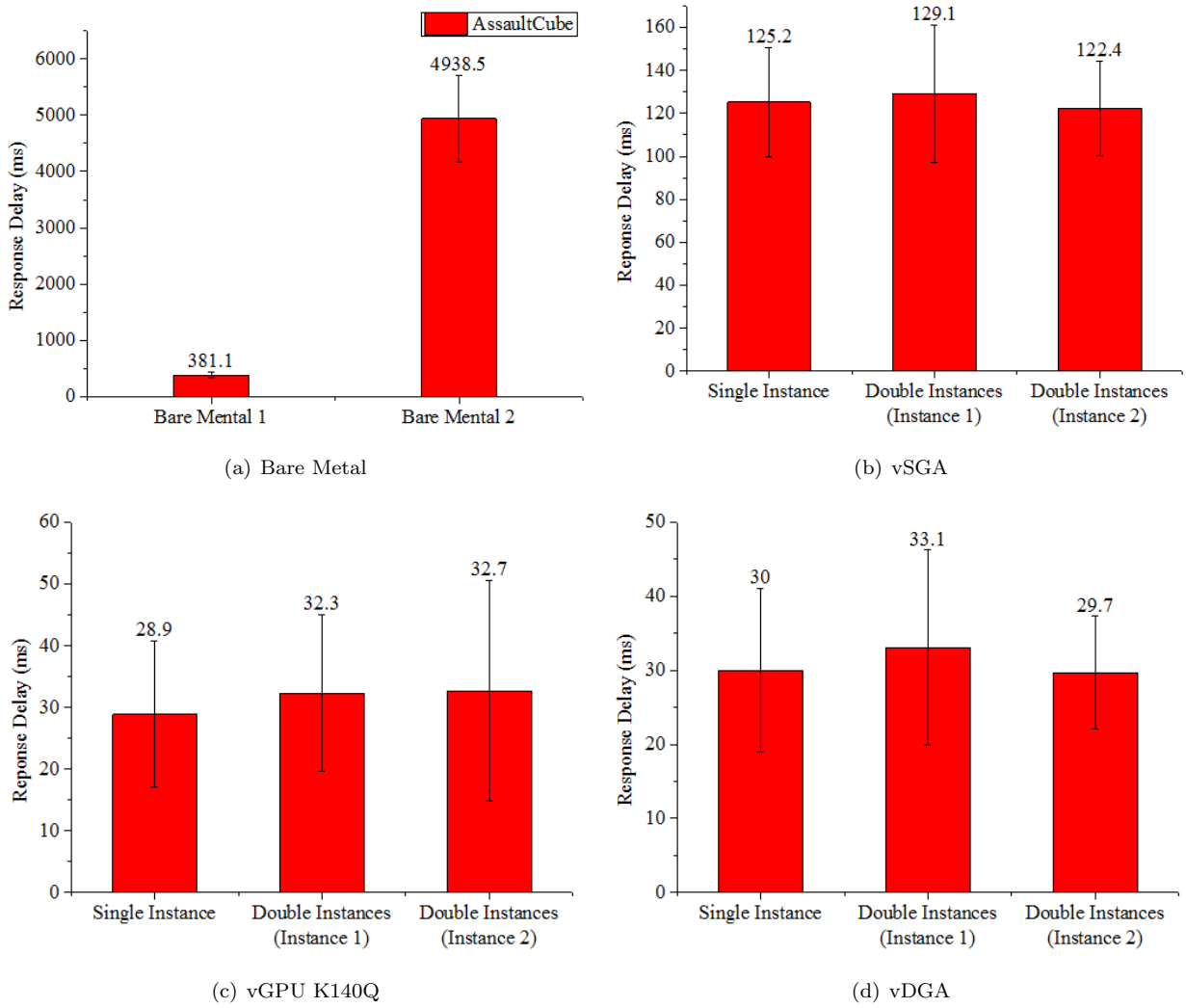


Figure 4.5: Unigine Sanctuary Performance: the vDGA Instances

it achieves the highest scores in the 2D benchmark of Fonts and Text among all tested configurations indicates it is more suitable for handling the software that falls into the use case of Knowledge Workers.

- All vGPU instances perform well in all experiments. They all achieve better performance than bare-metal system in all tests except the Doom 3 benchmark. In particular, they can gain much better results in the tests which the bare-metal system fails to deal with. For instance, bare-metal system 1 can only provide 381.1 ms on average as the response delay of AssaultCube (which is considered unacceptable), while all vGPU instances offer less than 33 ms on average as the response delay of this game. All in all, the results of all vGPU instances prove that vGPU technology is a promising technique for cloud gaming. In particular, as it provides multiple vGPU profiles, vGPU technology can be dynamically configured to meet the needs of various games. Nevertheless, the vGPU instance that has additional GPU resources does not achieve better performance in the experiments. This needs further analysis to discover the bottleneck of the performance degradation.
- vDGA achieves the best performance in all experiments. Additionally, it wins from the comparison between itself and vGPU K180Q. Although both of them are equipped with the same hardware resources, including graphic memory, CPU and RAM resources, and the same Nvidia CUDA cores, only the vDGA instance successfully utilizes them to gain better performance, especially when some advanced features of the graphics card are disabled. For example, when running without the advanced features of graphics card listed in Table 3.6, vDGA instance performs at 45.3 fps on average in the PassMark PerformanceTest 3D complex benchmark, while the vGPU K180Q only performs at 37.6 fps on average. This may be because



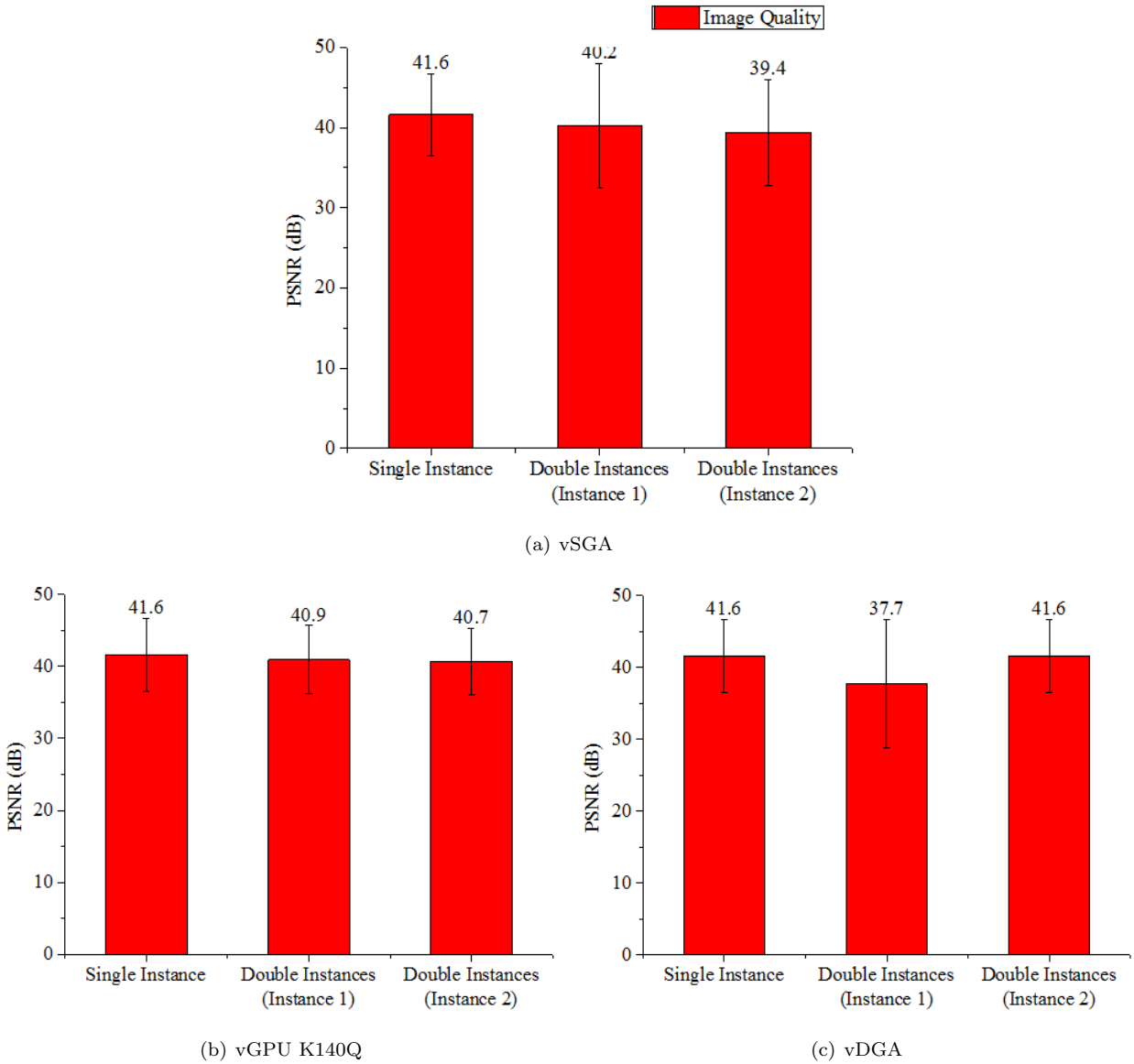
(a) Bare Metal

(b) vSGA

(c) vGPU K140Q

(d) vDGA

Figure 4.6: Response Delay of AssaultCube in Double Instances



(a) vSGA

(b) vGPU K140Q

(c) vDGA

Figure 4.7: Image Quality

vGPU technology still needs to communicate with physical graphics card via vGPU manager, while vDGA can directly talk to it, which reduces the overhead of communication. Nevertheless, although vDGA can achieve the best performance in the experiments, the price of using this technique to realize cloud gaming is expensive due to the fact that the GPU occupied by a vDGA instance cannot be shared by other VMs. Therefore, it is more wise to utilize this technique to deploy high-end games.

- The stability and potential scalability of each GPU virtualization solution are good. Firstly, each tested configuration in the experiments produce stable results (with low standard deviation) in all benchmarks. Secondly, when testing the potential scalability of each GPU virtualization solution, no matter if a single instance or the first double instance running on bare-metal system 1, or the second double instance running on bare-metal system 2, they all produce similar and stable results in all tests.

Chapter 5

Conclusions

VMware ESXi is one of the most popular virtualization platforms, and has been widely used in industry. It provides three GPU virtualization solutions for satisfying increasing commercial demand for full virtualization. Nevertheless, there is a lack of published work concerning the performance of these GPU virtualization solutions in cloud gaming. This thesis helps in evaluating the performance of the GPU virtualization solutions in VMware ESXi, including vSGA, vGPU and vDGA, by running graphics card benchmarks and measuring response delay and image quality of real cloud games. This chapter summarizes the experimental results and thesis contributions to the field of research, and possible future work.

5.1 Thesis Summary

There is no knowledge in the published literature about the performance of VMware GPU virtualization solutions in cloud gaming. It is important to evaluate the performance of these solutions in cloud gaming. Such evaluations can help cloud gaming vendors to know whether these solutions can fit the needs of cloud gaming, and to understand the advantages and disadvantages of each solution in cloud gaming.

Three graphics card benchmarks, including Unigine Sanctuary, PassMark PerformanceTest and Doom 3 benchmark, are utilized in the experiments to objectively evaluate the GPU performance in each tested configuration. There are several promising findings through the results. Firstly, the poor performance of vSGA in the Doom 3 benchmark and PassMark PerformanceTest 3D benchmark, and the fact that it does not support some high 3D features, such as Anti-aliasing, indicate vSGA may not be a good choice for cloud gaming. Nevertheless, vSGA achieves the highest scores in the 2D benchmark of Fonts and Text and Window Interface among all tested configurations, indicating vSGA is more suitable for handling the software which falls into the use case of Knowledge Workers. Secondly, all vGPU instances (vGPU K120Q, vGPU K140Q and vGPU K180Q) achieve better performance than the bare-metal system in the Unigine Sanctuary and Passmark PerformanceTest benchmarks in each configuration. Additionally, there is an interesting finding that all three vGPU instances achieve similar results in Unigine Sanctuary and Passmark PerformanceTest benchmarks, which means that the vGPU K140Q and vGPU K180Q fail to utilize the additional GPU resources they have. Thirdly, vDGA achieves the best performance among the three GPU virtualization solutions in all benchmarks. In particular, it performs better than vGPU K180Q even through they are

equipped with the same hardware resources, indicating that vDGA is the best GPU virtualization technology among the three solutions. Fourthly, the results show that each tested solution produces predictable results (with low standard deviation) in these benchmarks, which means the stability of each tested solution is excellent.

Three games representing three popular game genres (Avatar-First person, Avatar-Third person and Omnipresent) are utilized to measure whether these GPU virtualization solutions can produce acceptable image quality and response delay in cloud games. The vDGA and all vGPU instances provide excellent response delay for each tested game. Although vSGA produces unacceptable response delay in the game which represents first person shooter games, it provides acceptable response delays for the other two games. In addition, all three solutions achieve high image quality in the tested games.

For further assessing the potential scalability of each GPU virtualization solution, double instances of each tested solution are also evaluated in the experiments. The results show that the potential scalability of each tested solution is good. Regardless of whether a single-instance or double-instance scenario is deployed, each solution produces similar results in each tested benchmark, as well as similar image quality and response delay in each tested game.

Overall, the experiments performed in this thesis indicate that vGPU and vDGA are two promising GPU virtualization solutions in cloud gaming. These two solutions achieve good performance in the tested graphics card benchmarks, and provide excellent image quality and response delay in each tested game. The poor results in some benchmarks such as the Doom 3 benchmark and the fact that some 3D features are not supported in vSGA make it not suitable for cloud gaming scenarios.

5.2 Thesis Contributions

This thesis explores GPU pass-through and two GPU sharing virtualization solutions provided by VMware in cloud gaming scenarios. It makes the following contributions:

- This thesis is the first systematic study to evaluate VMware's GPU virtualization solutions in cloud gaming. The performance of each solution is measured using some graphics card benchmarks. Second, three cloud games are used to evaluate whether these solutions can guarantee acceptable image quality and tolerable response delay for each category of games. Third, the potential scalability of these solutions is evaluated by running two instances simultaneously.
- The analysis of the hardware resource consumption of each GPU virtualization solution is also performed. Additionally, the experiment measures the pattern of hardware resource consumption while launching each type of GPU virtualization instance.

5.3 Future Work

Future work can be divided into four broad categories: Further analysis of the hardware resource consumption of each GPU virtualization instance, further measurement of the potential scalability of each GPU virtualization solution, the study about each GPU virtualization's capability and performance regarding running high-end games and network influence on cloud games.

5.3.1 Further Analysis of the Hardware Resource Consumption

Further analysis of the hardware resource consumption in each GPU virtualization instance is necessary, as the work in this thesis only measures the consumption of four hardware resources and fails to find the bottleneck that stops some GPU instances which have more GPU resources, such as the vGPU K180Q and vDGA instances, from achieving better performance in some benchmark tests. The future work can use other performance monitoring tools, like Intel VTune Amplifier,[1] to gain more details of the hardware resource consumption. Moreover, the hot-spots of benchmarks can also be found with these tools, which can also help in finding the bottleneck of each GPU virtualization instance in each benchmark.

5.3.2 Further Measurement of the Potential Scalability

The potential scalability of each GPU virtualization technology requires further investigation since in the experiments only two instances are launched simultaneously. To prove the scalability of each GPU virtualization technology in large scale deployment, scenarios where more multiple simultaneous instances run the same benchmark/cloud game need to be tested.

5.3.3 The Capability and Performance of Running High-end Games

Although the experiments prove that vGPU and vDGA provide excellent image quality and acceptable response delay for all tested games, high-profile games were not tested. Future work can re-measure the image quality and response delay with some high-profile games, like Resident Evil 4: Ultimate HD Edition, which requires at least 2 GB of RAM and 15GB of hard disk space. Moreover, this game requires a GPU whose core speed and memory speed should be at least 600 MHz and 800 MHz respectively. This game would better stress the GPU virtualization instances.

5.3.4 Network Impact

In the experiments in this thesis, network conditions are essentially perfect as all the machines and VMs are connected with a private network, in which the network delay is below 1 ms and the packet loss rate is

[1] Intel VTune Amplifier Official Site. Website:https://software.intel.com/en-us/intel-vtune-amplifier-xe. Accessed: Dec 15, 2015.

close to 0%. Nevertheless, network delays and packet loss always exist in reality, which cause the increment of response delay and the loss of image quality respectively, degrading the user experience. Therefore, it is necessary to consider the impact of network conditions on each GPU virtualization solution. Future work can run the same tests under different network conditions.

References

[1] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I. Schoinas, R. Uhlig, B. Vembu, and J. Wiegert. Intel Virtualization Technology for Directed I/O. Technical report, Intel Inc., August 2006.

[2] M. Basanta. Reviewer's Guide for View in Horizon 6. Technical report, VMware Inc., 2015.

[3] F. Bellard. QEMU, A Fast and Portable Dynamic Translator. In *Proceedings of 2005 USENIX Annual Technical Conference*, pages 41–46, Anaheim, CA, April 2005.

[4] S. Campbell and M. Jeronimo. An Introduction to Virtualization. Technical report, Intel Corporation, 2006.

[5] Y. Chang, P. Tseng, K. Chen, and C. Lei. Understanding the Performance of Thin-client Gaming. In *Proceedings of 2011 IEEE International Workshop on Communications Quality and Reliability*, pages 1–6, Naples, FL, May 2011.

[6] K. Chen, Y. Chang, H. Hsu, D. Chen, C. Huang, and C. Hsu. On the Quality of Service of Cloud Gaming Systems. *IEEE Transactions on Multimedia*, 16:480–495, February 2014.

[7] K. Chen, Y. Chang, P. Tseng, C. Huang, and C. Lei. Measuring the Latency of Cloud Gaming Systems. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 1269–1272, Scottsdale, AZ, November 2011.

[8] S. Choy, B. Wong, G. Simon, and C. Rosenberg. The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-user Latency. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*, pages 2:1–2:6, Venice, Italy, November 2012.

[9] S. Chuah, C. Yuen, and N. Cheung. Cloud Gaming: A Green Solution to Massive Multiplayer Online Games. *IEEE Wireless Communications*, 21(4):78–87, August 2014.

[10] M. Claypool and K. Claypool. Latency and Player Actions in Online Games. *Communications of the ACM*, 49(11):40–45, November 2006.

[11] M. Claypool, D. Finkel, A. Grant, and M. Solano. Thin to Win?: Network Performance Analysis of the OnLive Thin Client Game System. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*, pages 1:1–1:6, Venice, Italy, November 2012.

[12] F. Diard. Cloud Gaming & Application Delivery with NVIDIA GRID Technologies. Technical report, NVIDIA Inc., 2014.

[13] Y. Dong, J. Dai, Z. Huang, H. Guan, K. Tian, and Y. Jiang. Towards High-quality I/O Virtualization. In *Proceedings of The 2nd Israeli Experimental Systems Conference*, pages 12:1–12:8, Haifa, Israel, May 2009.

[14] M. Dowty and J. Sugerman. GPU Virtualization on VMware's Hosted I/O Architecture. *SIGOPS Operating Systems Review*, 43(3):73–82, July 2009.

[15] J. Duato, A.J. Pena, F. Silla, J.C. Fernandez, R. Mayo, and E.S. Quintana-Orti. Enabling CUDA Acceleration Within Virtual Machines Using rCUDA. In *Proceedings of the 18th International Conference on High Performance Computing*, pages 1–10, Bangalore, India, December 2011.

[16] J. Duato, A.J. Pena, F. Silla, R. Mayo, and E.S. Quintana-Orti. Performance of CUDA Virtualized Remote GPUs in High Performance Clusters. In *Proceedings of the 40th International Conference on Parallel Processing*, pages 365–374, Taipei, Taiwan, September 2011.

[17] P. Eisert and P. Fechteler. Low Delay Streaming of Computer Graphics. In *Proceedings of the 15th IEEE International Conference on Image Processing*, pages 2704–2707, San Diego, CA, October 2008.

[18] G. Giunta, R. Montella, G. Agrillo, and G. Coviello. A GPGPU Transparent Virtualization Component for High Performance Computing Clouds. In *Proceedings of the 16th International Euro-Par Conference on Parallel Processing: Part I*, pages 379–391, Ischia, Italy, September 2010.

[19] V. Gupta, A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar, and P. Ranganathan. GViM: GPU-accelerated Virtual Machines. In *Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing*, pages 17–24, Nuremburg, Germany, March 2009.

[20] J.G. Hansen. Blink: Advanced Display Multiplexing for Virtualized Applications. In *Proceedings of the 17th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 1–6, Urbana-Champaign, IL, June 2007.

[21] H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu. Placing Virtual Machines to Optimize Cloud Gaming Experience. *IEEE Transactions on Cloud Computing*, 3:42–53, January 2015.

[22] H. Hong, T. Fan-Chiang, C. Lee, K. Chen, C. Huang, and C. Hsu. GPU Consolidation For Cloud Games: Are We There Yet? In *Proceedings of 13th Annual Workshop on Network and Systems Support for Games*, pages 1–6, Nagoya, Japan, December 2014.

[23] C. Huang, D. Chen, C. Hsu, and K. Chen. GamingAnywhere: An Open-source Cloud Gaming Testbed. In *Proceedings of the 21st ACM International Conference on Multimedia*, pages 827–830, Barcelona, Spain, October 2013.

[24] C. Huang, C. Hsu, Y. Chang, and K. Chen. GamingAnywhere: An Open Cloud Gaming System. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 36–47, Oslo, Norway, March 2013.

[25] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hossfeld. An Evaluation of QoE in Cloud Gaming Based on Subjective Tests. In *Proceedings of the 5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 330–335, Seoul, Korea, June 2011.

[26] A.E. Johnsen. VMware Horizon 6 3D Engineering Workloads Reference Architecture. Technical report, VMware Inc., 2015.

[27] A. Jurgelionis, P. Fechteler, P. Eisert, F. Bellotti, H. David, J.P. Laulajainen, R. Carmichael, V. Poulopoulos, and A. Laikari. Platform for Distributed 3D Gaming. *International Journal of Computer Games Technology*, 2009:1:1–1:15, January 2009.

[28] H. A. Lagar-Cavilla, N. Tolia, M. Satyanarayanan, and E. de Lara. VMM-independent Graphics Acceleration. In *Proceedings of the 3rd International Conference on Virtual Execution Environments*, pages 33–43, San Diego, CA, June 2007.

[29] P. Lee. NVIDIA GRID VGPU Deployment Guide for VMware Horizon 6.1. Technical report, NVIDIA Inc., 2015.

[30] Y. Lee, K. Chen, H. Su, and C. Lei. Are All Games Equally Cloud-gaming-friendly?: An Electromyographic Approach. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*, pages 3:1–3:6, Venice, Italy, November 2012.

[31] J. Liu, W Huang, B. Abali, and D.K. Panda. High Performance VMM-bypass I/O in Virtual Machines. In *Proceedings of 2006 USENIX Annual Technical Conference*, pages 3–17, Boston, MA, May 2006.

[32] D. Marshall. Understanding Full Virtualization, Paravirtualization, and Hardware-assist Virtualization. Technical report, VMware Inc., 2007.

[33] S. Nanda and T. Chiueh. A Survey on Virtualization Technologies. Technical report, Department of Computer Science, SUNY at Sony Brook, New York, 2005.

[34] J. Park, Y. Park, and S. Mahlke. Chimera: Collaborative preemption for multitasking on a shared gpu. In *Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 593–606, Istanbul, Turkey, March 2015.

[35] A. Puri, X. Chen, and A. Luthra. Video Coding Using the H. 264/MPEG-4 AVC Compression Standard. *Signal Processing: Image Communication*, 19(9):793–849, October 2004.

[36] R. Shea and J. Liu. On GPU Pass-Through Performance for Cloud Gaming: Experiments and Analysis. In *Proceedings of the 12th Annual Workshop on Network and Systems Support for Games*, pages 6:1–6:6, Denver, CO, December 2013.

[37] R. Shea, J. Liu, E.C.-H. Ngai, and Y. Cui. Cloud Gaming: Architecture and Performance. *IEEE Network*, 27(4):16–21, July 2013.

[38] L. Shi, H. Chen, and J. Sun. vCUDA: GPU Accelerated High Performance Computing in Virtual Machines. In *Proceedings of the 23th IEEE International Symposium on Parallel Distributed Processing*, pages 1–11, Rome, Italy, May 2009.

[39] C. Smowton. Secure 3D Graphics for Virtual Machines. In *Proceedings of the 2nd European Workshop on System Security*, pages 36–43, Nuremburg, Germany, April 2009.

[40] I. Tanasic, I. Gelado, J. Cabezas, A. Ramirez, N. Navarro, and M. Valero. Enabling preemptive multiprogramming on gpus. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, pages 193–204, June 2014.

[41] A.S. Tanenbaum and H. Bos. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, 4th edition, March 2014.

[42] K. Tian, Y. Dong, and D. Cowperthwaite. A Full GPU Virtualization Solution with Mediated Pass-through. In *Proceedings of 2014 USENIX Annual Technical Conference*, pages 121–132, Philadelphia, PA, June 2014.

[43] M.S. Vinaya, N. Vydyanathan, and M. Gajjar. An Evaluation of CUDA-enabled Virtualization Solutions. In *Proceedings of the 2nd International Conference on Parallel Distributed and Grid Computing*, pages 621–626, Himachal Pradesh, India, December 2012.

[44] J.P. Walters, A.J. Younge, D.I. Kang, K.T. Yao, M. Kang, S.P. Crago, and G.C. Fox. GPU Passthrough Performance: A Comparison of KVM, Xen, VMWare ESXi, and LXC for CUDA and OpenCL Applications. In *Proceedings of the 7th IEEE International Conference on Cloud Computing*, pages 636–643, Anchorage, AK, June 2014.

[45] Y. Wang. Survey of Objective Video Quality Measurements. Technical report, February 2006.

[46] Y. Wang, Y. Zhang, and J. Ostermann. *Video Processing and Communications*. Prentice Hall PTR, Upper Saddle River, NJ, 1st edition, October 2001.

[47] Z. Wang, L. Lu, and A.C. Bovik. Video Quality Assessment Based on Structural Distortion Measurement. *Signal processing: Image communication*, 19:121–132, February 2004.

[48] M. Yu, C. Zhang, Z. Qi, J. Yao, Y. Wang, and H. Guan. VGRIS: Virtualized GPU Resource Isolation and Scheduling in Cloud Gaming. In *Proceedings of the 22nd International Symposium on High-performance Parallel and Distributed Computing*, pages 203–214, New York, NY, June 2013.

[49] C. Zhang, J. Yao, Z. Qi, M. Yu, and H. Guan. vGASA: Adaptive Scheduling Algorithm of Virtualized GPU Resource in Cloud Gaming. *IEEE Transactions on Parallel and Distributed Systems*, 25(11):3036–3045, November 2014.