ISSN 2347 - 3983
Volume 7, No. 9 September 2019
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter02792019.pdf
https://doi.org/10.30534/ijeter/2019/02792019

A Desktop as a Service Based on GPU Pass-Through

Sang Boem Lim
Department of Smart ICT Convergence and Social Eco-Tech Institute, Konkuk University, Seoul, Korea,
sblim@konkuk.ac.kr

ABSTRACT

Nowadays, many vendors that provide virtualization solutions also provide commercial desktop as a service (DaaS) to cloud users. However, these open source virtualization solutions have graphics performance limitation issues associated with 3D rendering. In this paper, we present an implementation of GPU virtualization using GPU pass-through technology to enable high graphics performance by KVM hypervisor based VMs. We focus on GPU virtualization, specifically, GPU direct pass-through, as a means of providing 3D rendering for cloud DaaS service to game users. We use the KVM hypervisor as our background system virtualization technology.

Key words : Desktop as a Service, GPU, Pass-through, Graphics Performance

1. INTRODUCTION

The typical solutions of Desktop as a Service (DaaS) are Citrix XenDesktop [1] and VMware vSphere Desktop [2]. These DaaS services are provided based on XenServer, vShpere, and VMware proprietary virtualization solutions. In contrast, KVM [3] and Xen [4] hypervisors also contain desktop virtualization technology but are open source. Open source cloud management platforms (such as OpenStack, CloudStack, and OpenNebula) have been developed to provide cloud services, most of which are provided based on open source KVM or Xen hypervisors.

Handling GPU as a peripheral component interconnect (PCI) device is the most appropriate way to virtualize GPU. However, a GPU has its own specificity related to bios information, which varies according to vendor, so it is difficult to virtualize. There are several GPU virtualization technologies, such as gVirtuS [5], and GVim [6]. These technologies have been developed to implement sharing of GPU resources among VMs. In order to share GPU resources, the hypervisor must satisfy all requirements of the GPU API in accordance with different versions of applications such as CUDA [7] and OpenCL [8]. When there is a change in API version, the GPU API also needs to change, which means that it has low portability. Sharing one graphics card among several VMs via the time-sharing technique results in performance degradation. In addition, according to the different hypervisors' scheduling algorithms, delay time will also be introduced. In this study, we used PCI direct pass-through technology to increase portability and performance. Subsequently, we compared and analyzed bare-metal graphics performance to that of the GPU pass-through VM using the SpecViewperf [9] and Unigine Heaven [10] benchmark tools.

The remainder of this paper is organized as follows. Section 2 gives an overview of cloud technology, including cloud DaaS service, hypervisor, and GPU virtualization. Section 3 outlines the test-bed system design including graphics cards tested, hardware, and benchmark software. Section 4 describes the experiments conducted and analyzes the performance results obtained from two benchmark tools for each graphics feature. Section 5 concludes this paper and outlines plans for further research.

2. CLOUD TECHNOLOGY OVERVIEW

2.1 Cloud DaaS Service

The cloud typically has three service layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In Figure 1, IaaS is the service based on infrastructure, which is at the network architecture level. OpenStack, OpenNebula, and CloudStack are the most popular private cloud IaaS solutions. PaaS is a service based on platform, which is used by application developers. It provides a development environment to developers, which enables them to concentrate on coding without worrying about the environment. Google App Engine is a well-known PaaS service. SaaS is a software-based service and is utilized by many users. Google Apps can be used as a solution to implement SaaS. With a Google account, anyone can use Google Apps through the Internet without installing Google Apps.

In addition to these service types, many other services are provided by cloud service providers (CSPs). For example, DaaS, Database as a Service (DBaaS), Firewall as a Service (FaaS), and load balancer as a service (LBaaS). Among these, depending on the increasing needs of the Windows environment gaming or graphics users, DaaS has become

progressively more popular than any other service. Citrix XenDesktop is one of the most popular commercial virtual desktop applications. It provides an accessible pay-as-you-go model to scale up in busy seasons and reduce expenditures when necessary. XenDesktop uses XenServer with most use cases commercially supported by Citrix.

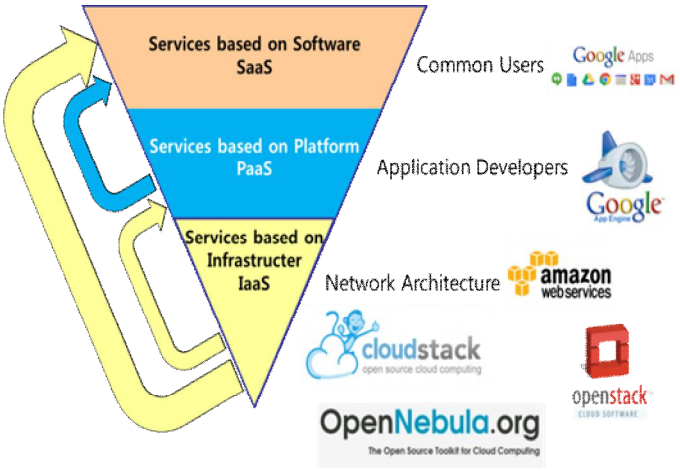

Figure 1: Cloud service types

2.2 GPU Direct Pass-through

GPU direct pass-through technology can minimize interaction from the hypervisor to enable bare-metal performance when VMs access the GPU. It uses I/O memory management unit (IOMMU) virtualization provided by hardware. When an OS is running in a VM, it does not usually know the physical addresses of the host memory that it accesses. This makes it difficult to send direct memory access (DMA) commands to peripheral hardware and causes a delay in the I/O operation. IOMMU solves this problem by mapping guest-physical addresses to host-physical addresses.

In order to implement GPU direct pass-through, the Linux kernel has to be recompiled to a recent version (3.6+) that enables support for DMA remapping devices, enable DMA remapping devices, PCI stub driver, and support for interrupt remapping. In this study, we activated the DMA function that enables GPU direct pass-through. DMA can make hardware, graphics cards, and network cards directly access memory, and thereby improve performance. KVM uses QEMU, developed as open source software, as its virtual emulator.

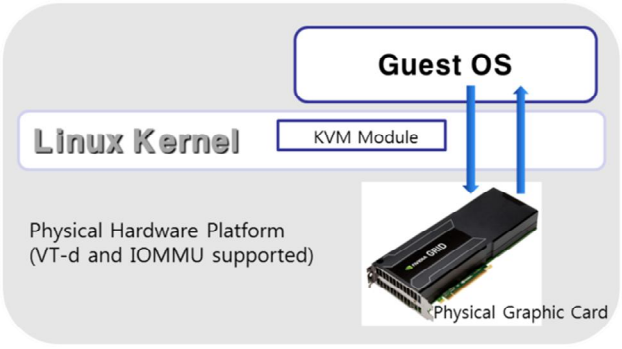

Figure 2: PCI pass-through on KVM

The Linux kernel already includes the kvm.ko kernel module. QEMU has several options such as vga cirrus, vga std, vga vmware, and vga qxl for emulating a graphics card. Vga cirrus is the most commonly used graphics emulation option; consequently, each guest OS has its own built-in driver inside. Vga std is only supported for resolutions higher than 1080 × 1024 × 16 and only the newest version of Linux and Windows has built-in drivers for it. Vga vmware requires the VMware SVGA-II driver for each guest OS and Linux guest OSes need to install x11-drivers/xf86-video-vmware. When higher graphics performance is needed, the vga qxl option, which uses the SPICE remote display environment protocol, can be used. These options emulate the host graphic card, which causes performance degradation. Furthermore, most options do not support the video virtualization function. In order to run applications such as CAD, which uses the OpenGL library, the OpenGL function has to be enabled to provide high graphics performance. Therefore, it is necessary to consider graphics card as a PCI device to which we can directly attach a host graphic card to the VM. Figure 2 illustrates PCI pass-through on KVM. The hardware must support Intel vt-d—a direct I/O virtualization technology—and IOMMU.

3. SYSTEM DESIGN

We designed a test-bed system containing test graphics cards, hardware, and benchmark software. Figure 3 shows the experimental system configuration. We constructed three hosts: host 1 for testing using a bare-metal Windows 7 environment, host 2 for testing KVM-based Grid K1 pass-through performance with CentOS 6.5 Linux kernel version 3.13, and host 3 for testing KVM-based Grid K2 pass-through performance with CentOS 6.5 Linux kernel version 3.13.

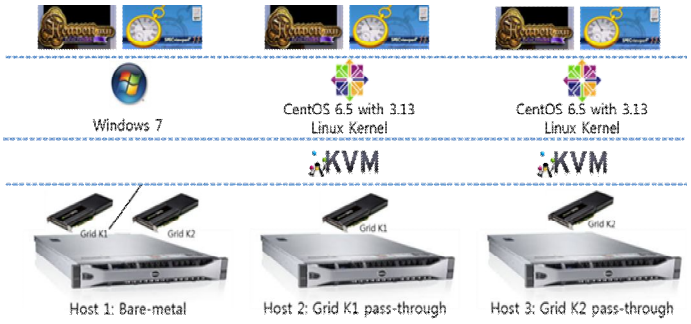

Figure 3: System configuration

3.1 Tested Graphic Cards

Use In our implementation of GPU direct pass-through, we tested NVIDIA Grid K1 and Grid K2 graphic cards. We used NVIDIA's Kepler based GRID K1 and K2 boards. They also work closely with leading server vendors such as Cisco, Dell, HP, IBM, and SuperMicro.

Table 1 compares the specifications of Grid K1 and Grid K2 [11]. The results of our comparison for the two graphics cards indicate that Grid K2 has a higher performance than Grid K1. However, because neither has a display I/O, we used the remote desktop technology to provide remote access to the guest OS, and then executed the benchmark applications. Considering the performance degradation caused by remote access, after executing our applications, we disabled remote connection and then checked the result after several minutes had elapsed.

Table 1: Comparison of Grid K1 and Grid K2 specifications

	Grid K1	**Grid K2**
Number of GPUs	4 x entry Kepler GPUs	2 x high-end Kepler GPUs
Total NVIDIA CUDA cores	768	3072
Total memory size	16GB DDR3	8GB GDDR5
Max power	130 W	225 W
Board width	Dual slot	Dual slot
Display I/O	None	None
Aux power	6-pin connector	8-pin connector
PCIe	x16	x16
PCIe generation	Gen 3	Gen 3

3.2 Hardware Information

Table 2 shows the bare-metal hardware information for our test-bed host 1 in Figure 7. We used a Dell PowerEdge R720 server board, Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz with VT-d CPU, 12 cores, 132 GB RAM, and NVIDIA Grid K1/K2 graphics card with driver version 347.25.

Table 2: Bare-metal hardware environment information for test-bed host 1

Bare-metal Environment Information	
Server Board	Dell PowerEdge R720
CPU	Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz with VT-d
CPU Numbers	12
Memory	132 GB
Graphic Card	NVIDIA Grid K1/Grid K2
Host OS	64bit Windows 7
Driver Version	347.25

Table 3 shows the pass-through environment information for our test-bed hosts 2 and 3 in Figure 7. We created only one VM on host 2, while host 3 as similar capacity to that of a bare-metal host. We used Dell PowerEdge R720 for the host server board, Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz with VT-d CPU, 12 host cores (11 cores for VMs, and one for the hypervisor and the running system), 132 GB host RAM (130 GB for VM memory and 2 GB for hypervisor and running system), KVM with QEMU version 1.3, CentOS 6.5 with Linux kernel 3.13 as the host OS, 64 bit Windows 7 as the guest OS, and NVIDIA Grid K1/K2 graphics card with driver version 347.25 for the guest OS.

Table3: Pass-through environment information

Pass-through Environment Information	
Server Board	Dell PowerEdge R720
Host CPU	Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz with VT-d
Host CPU numbers	12
VM CPU numbers	11
Host Memory	132 GB
VM Memory	130 GB
Hypervisor	KVM QEMU 1.3 version
Host OS	CentOS 6.5 with Linux kernel 3.13
Guest OS	64 bit Windows 7
Graphic Card	NVIDIA Grid K1/Grid K2
Driver Version	347.25

4. EXPERIMENTAL RESULTS AND ANALYSIS

Figure 4 shows the results obtained using SpecViewperf 11. We compared K1 bare-metal, K1 pass-through, K2 bare-metal, and K2 pass-through performances in terms of frames/sec. In addition, we calculated pass-through

performance for each grid graphics card by percentage value. The resulting effect depends on the character of the benchmark models.

For the ensight-04 and snx-01 benchmark models, K2 exhibits twice the speed performance of K1. The performance of K1 pass-through is 13.94 frames/sec, which is 97.5% that of K1 bare-metal performance on the ensight-04 model. The performance of K2 pass-through is 56 frames/sec, which is 98.5% that of K2 bare-metal performance on the ensight-04 model. The performance of K1 pass-through is 17.98 frames/sec which is 97% that of K1 bare-metal performance on the snx-01 model. The performance of K2 pass-through is 45.08 fames/sec, which is 99.1% that of K2 bare-metal performance on the snx-01 model.

For the proe-05 benchmark model, both K1 and K2 exhibit poor performance. The performance of K1 pass-through is 5.14 frames/sec, which is only 46.2% that of K1 bare-metal performance. The performance of K2 pass-through is 5.4 frames/sec, which is only 42% that of K2 bare-metal performance. Thus, it is clear that this model has low graphics performance.

The results of this experiment showed that ensight-04 and snx-01 both have large vertices ranges. From comparison of K1 and K2 specifications, it can be seen that K2 has 3072 cores, which is more than that possessed by K1. For this reason, as progressively more vertices are needed, K2 exhibits higher performance than K1 on snx-01 because it utilizes a multi-pass stencil-based algorithm. Furthermore, the pore-05 model contains the TexGen function—a geometric textile modeling software package containing mixed 3D and 2D features. TexGen can be exported to the CAD exchange file formats IGES and STEP. Consequently, time is required to prepare for modeling so that when it is used by CAD no further time is consumed for preparation. In addition, the pore-05 model has its vertices in the range 7 million to 13 million, the smallest range in SpecViewperf. These results indicate that when graphics intensive programs such as CAD or 3D animations are executed, larger vertices numbers result in better GPU performance.

In the Heaven benchmark tool, the rendering API can be set as Direct3D 11, Direct3D 9, or OpenGL. We disabled tessellation and anti-aliasing and set the resolution mode to 1280 × 1024, which is the same as that of SpecViewperf used in the test. Direct3D 11 has advanced enhanced tessellation and anti-aliasing. Therefore, in order to compare the performance of the same functions from the three rendering APIs, we disabled the tessellation and anti-aliasing functions.

In Figure 9, the performance of K1 pass-through using the Direct3D 11 rendering API is 46.64 frames/sec, which is 81.3% that of K1 bare-metal. The performance of K2 pass-through for Direct3D 11 is 41.8 frames/sec, which is only 59.4% that of K2 bare-metal performance. In this case, we discovered that the virtualization performance of K1 is better than that of K2, whereas in terms of graphics card bare-metal performance, K2 is much better than K1.

With the Direct3D 9 rendering API, we found that the total performance of K1 and K2 is less than when the Direct3D 11 rendering API is used. In this case, it is clear that even though both rendering APIs are specialized to Microsoft Windows OSes, Direct3D 11 was released for Windows 7 and therefore has no support for XP. The most important improvement of Direct3D 11 is tessellation; in addition, its performance also has improved more than 30%. Consequently, when we have low graphic performance capacity and also need to run on Windows XP OS, it is better to use Direct3D 9. Otherwise, it is better to use Direct3D 11, because it has more enhanced performance improvement and is designed to be specialized to high performance capacity only running on the Windows 7 OS.

For the OpenGL rendering API, we found that the performance of K1 is less than that for Direct3D 9. Otherwise, the performance of K1 pass-through is 11.54 frames/sec, which is almost 100% that of K1 bare-metal. In addition, the performance of K2 pass-through is 29.44 frames/sec, which is 50.2% that of K2 bare-metal and higher than Direct3D 9 performance. In this case, even though the performance of K2 bare-metal is sufficiently high, the virtualization performance

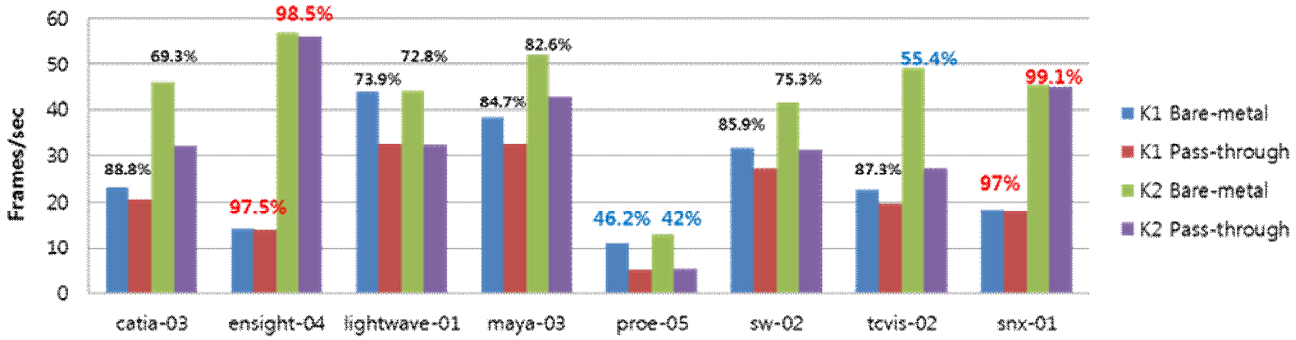

Figure 4: Test results using SpecViewperf 11

of K2 pass-through is only 50.2%. The performance of Direct3D 11 is better than that of OpenGL because OpenGL can be used on cross platforms such as Windows, Linux, and Mac OS environments, whereas Direct3D is specialized to Windows OS.

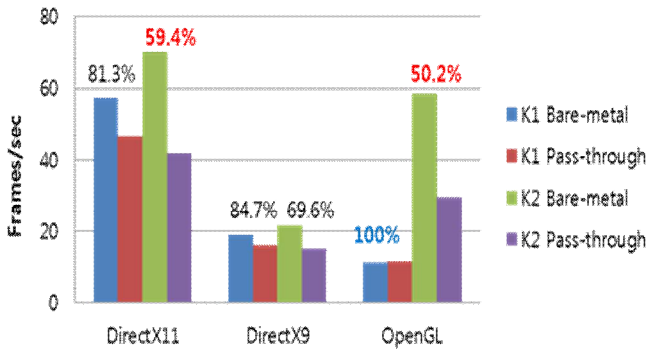

Figure 9: Test results using Unigine Heaven 4

Graphics performance is complex and diverse and depends on the various graphic vendors and capacity. The results of this performance test comparison indicate the following: 1) application of direct pass-through does not guarantee complete bare-metal performances but can be close; 2) depending on the number of cores and vertices tested, virtualization performance can be much higher; 3) according to the characteristics of the benchmark models, suitable rendering APIs can be used for better performance; 4) in order to ensure maximum virtualization performance, graphics card vendors, rendering APIs, and parallel algorithms need to be investigated.

5. CONCLUSION

In this paper, we reported on an implementation and performance analysis of GPU direct pass-through for cloud DaaS service based on the KVM hypervisor. We used two graphics intensive benchmark tools to test graphics and virtualization performances. Depending on several comparison works, we analyzed factors that affect performance. Virtualization is a system level technical issue in the cloud research area. Desktop virtualization is one of the hottest topics and graphics virtualization is one of the most important issues. In order to ensure maximum GPU virtualization performance, more research and trial services are needed.

In the future, consequent on this study, we intend to develop an automated GPU pass-through deployment environment using the OpenStack cloud management platform so that ordinary cloud users can easily deploy their own virtual desktop environments.

REFERENCES

1. Citrix. **Citrix Virtual Apps and Desktops 7 1906 product documentation**, 2019. (available at https://docs.citrix.com/)
2. Y. J. TONG, W. Q. YAN, and J. YU. **Analysis of a secure Virtual Desktop Infrastructure system**, *International Journal of Digital Crime and Forensics*, vol. 7, no 1, pp. 69-84, Jan. 2015. https://doi.org/10.4018/IJDCF.2015010104
3. C. Jiang, Y. Wang, D. Ou, Y. Li, J. Zhang, J. Wan, B. Luo, and W. Shi. **Energy efficiency comparison of hypervisors**, *Sustainable Computing: Informatics and Systems*, vol. 22, pp. 311-321, Jun. 2019. https://doi.org/10.1016/j.suscom.2017.09.005
4. M. Korniichuk, K. Karpov, I. Fedotova, V. Kirova, N. Mareev, D. Syzov, and E. Siemens. **Impact of Xen and Virtual Box Virtualization Environments on Timing Precision under Stressful Conditions,** in Proc. MATEC Web of Conferences, 2018, art. no. 02006. https://doi.org/10.1051/matecconf/201820802006
5. R. Montella, G. Giunta, G. Laccetti, M. Lapegna, C. Palmieri, C. Ferraro, and D. S. Nikolopoulos. **On the virtualization of CUDA based GPU remoting on ARM and X86 machines in the GVirtuS framework**, *International Journal of Parallel Programming*, vol. 45, no. 5, pp. 1142-1163, Oct. 2017. https://doi.org/10.1007/s10766-016-0462-1
6. GVim Portable, http://portablegvim.sourceforge.net/ (Accessed on Aug. 20, 2019)
7. L. Jian, C. Wang, Y. Liu, S. Liang, W. Yi, and Y. Shi. **Parallel data mining techniques on Graphics Processing Unit with Compute Unified Device Architecture (CUDA)**, *Journal of Supercomputing*, vol. 64, no. 3, pp. 942-967. Jun. 2013. https://doi.org/10.1007/s11227-011-0672-7
8. Khronos OpenCL Working Group. **The OpenCL Specification, version 2.2**, 2017.
9. SepcViewperf, https://www.spec.org/ (Accessed on Aug. 22, 2019)
10. Unigine Heaven, https://unigine.com/ (Accessed on Sep. 2, 2019)
11. Nvidia, https://nvidea.com/ (Accessed on Sep. 6, 2019)