

東海大學

資訊工程研究所

碩士論文

指導教授: 劉榮春博士, 楊朝棟博士

於 OpenStack 上實作具 GPU 加速的虛擬桌面基
礎設施

The Implementation of a Virtual Desktop
Infrastructure with GPU Accelerated on
OpenStack

研究生: 李政岳

中華民國一零六年六月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 李 政 岳 所提之論文

於 OpenStack 上實作具 GPU 加速的虛擬桌面
基礎設施

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 許慶賢 簽章

委 員 黃國辰

姜再強

指 導 教 授 楊朝棟 簽章

指 導 教 授 劉榮春 簽章

中華民國 106 年 6 月 27 日

摘要

這幾年，資訊產業發展趨勢逐漸邁向雲端，各種雲端相關的技術與應用與日俱增，隨著雲端技術的普及，無論是企業用戶或是個人用戶使用雲服務的數量也有爆炸性的提升。而雲端虛擬桌面則為使用者最先接觸到的一個部份，而雲端虛擬桌面的性能則與網路以及 GPU 有所牽連，如何提供一個順暢的雲端虛擬桌面就是一個很重要的議題，本論文提出了在 OpenStack 上整合圖形處理器 (Graphics Processing Unit, GPU)，並且透過 PCI Pass-Through 的方式，來解決虛擬機顯示效能的問題，透過將 GPU 資源分配至虛擬機，讓虛擬機器達到圖形處理加速的能力，實驗的部份將會透過不同的基準測試軟體來測量虛擬機，在三種不同的驅動程式其中包括了 OpenGL、DirectX11 以及 DirectX9 進行測試，另外也因為在雲端環境中隨需自助服務的特性，本文也會將 vCPU 的因素也納入考量，在實驗中調整不同數量的 vCPU 進行實驗，藉此測量出在不同情況下的虛擬機 FPS(Frame per Second, FPS, 畫面幀率) 藉此評斷虛擬機的效能。透過本文的實驗發現三個不同的驅動程式對於虛擬桌面畫面處理的能力也不盡相同，其中由 DirectX11 的效能最好，且會因為 vCPU 的數量增加而增加顯示的效能，在 16 個 vCPU 的情況下 DirectX11 達到最高的圖形處理能力，在 Heaven Benchmark 中 Max FPS 遠遠大於 DirectX9 以及 OpenGL 約兩倍左右。而平均 FPS 的部分，DirectX11 大於 OpenGL 約兩倍，大於 DirectX 約 3 倍左右，透過這些數據可以發現 DirectX11 與 NVIDIA 的 K2 GPU 相容性最高，對虛擬桌面的畫面處理能力提升最多。另外本文在實驗中也發現 DirectX9 與 vCPU 的數量似乎沒有相關性，而 OpenGL 與 DirectX11 兩個都會因為 vCPU 的數量而有所改變。

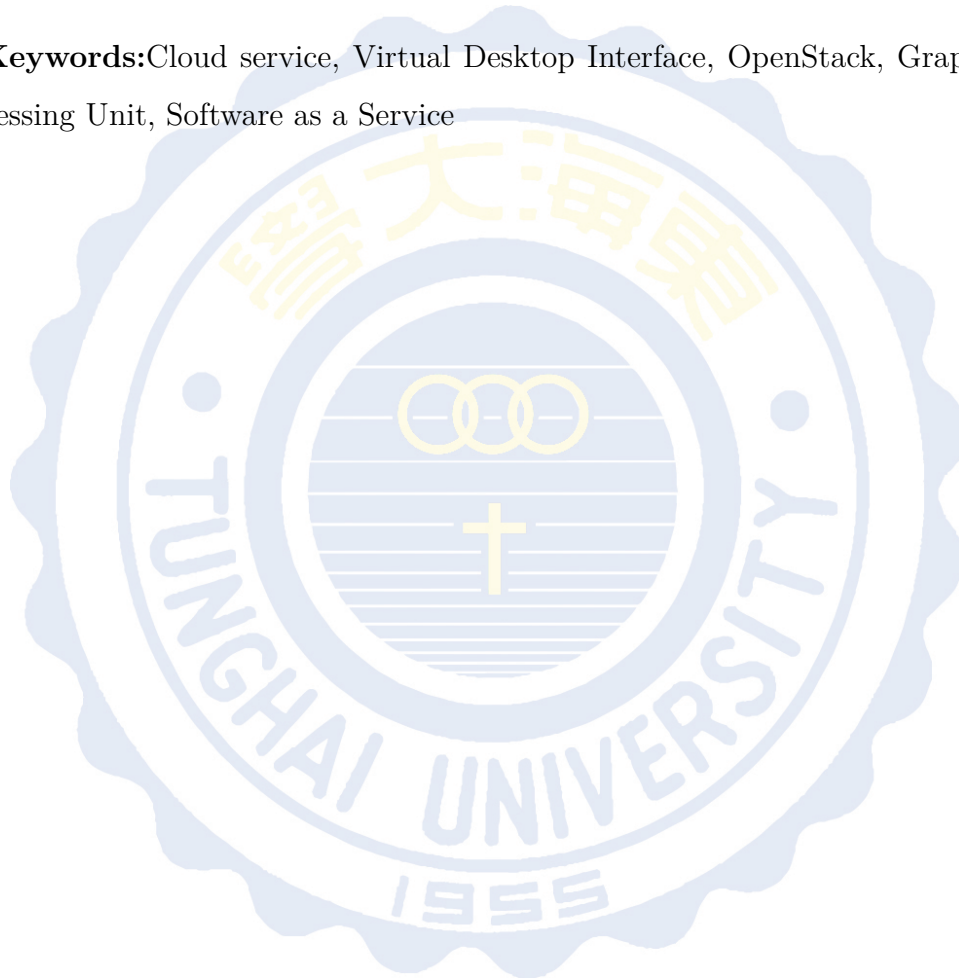
關鍵字: 雲端服務，虛擬桌面，OpenStack，圖形處理器，軟體即服務

Abstract

In recent years, Information industry trends gradually towards the clouds. Various clouds are related technologies and applications are growing with the popularity of cloud technology. Whether the number of enterprise users or individual users use cloud services is also an explosive upgrade. While cloud virtual desktop is the first user access part of the cloud and the performance of cloud virtual desktop with the network and GPU has been implicated. Therefore, a smooth cloud virtual desktop is a very important issue in this environment. This study proposed an integration of GPU on OpenStack and through PCI Pass-Through way to solve the problem of virtual machine display performance by allocating GPU resources for virtual machine. So, the virtual machine can achieve the ability of graphic acceleration. For the experimental, we will use different benchmarking software to measure the virtual machine. Among three different drivers that consist of OpenGL, DirectX11 and DirectX9 for testing and also because of the characteristics of on-demand self-services on the cloud environment, this study uses vCPU factor as experiment considering. In the experiment, adjust the different number of vCPUs to measure the different cases performance of FPS virtual machine (Frame per Second). Through this experimental, we found out there are three different processing drivers for virtual desktop. Under 16 cores of virtual CPU condition, DirectX11 is the best performance. In the Heaven Benchmark environment, Max FPS is twice better performance than DirectX9 and OpenGL. The average of FPS values, Direct X11 is twice better performance than OpenGL, three times better performance than DirectX. Through these data found out DirectX11 and NVIDIA

's K2GPU have highest compatibility. The virtual desktop graphics processing capacity enhance the most. Besides that, our experiments found out the number of DirectX9 and vCPU seems no correlation. Then, because of the number of vCPU factor, OpenGL and DirectX11 have changed.

Keywords:Cloud service, Virtual Desktop Interface, OpenStack, Graphics Processing Unit, Software as a Service



致謝詞

在研究所的這兩年期間，兩年說長不長，說短不短，在這段時間學習的過程之中，每分每秒都是特別的重要。首先我要感謝我的指導教授楊朝棟博士，在兩年期間給了我許多不管是實質上的，包括是設備或是其他實體的幫助，亦或者是精神上的指導幫助，給了我許多的建議，讓我在研究的方向能更加的明確。並且讓我有機會參加各式各樣的研討會或是競賽，讓我從中學習獲取經驗，得到更多的專業知識以及磨練技術。也因為這些經驗的累積，才讓我能夠順利的完成本篇論文。謹此致最深之謝忱，讓我不僅在獨立思考、上台發表以及執行專案能力都更加進步。

除此之外我還要感謝我的實驗室同學、以及不管是研究所的學弟、妹們或是大學部的學弟、妹。謝謝你們在我需要幫忙的時候對我伸出援手，協助我完成各種艱難的挑戰。同時也要感謝已經畢業的學長們，即使畢業也還是持續的幫助我，協助我完成各式各樣的專案，以及給了我許多建議，不管是實驗方面亦或者是人生的建議。讓我能從一個什麼都不懂的狀況，跟著大家一路走到今天。現今已能獨立完成專案、上台發表都更加的進步。

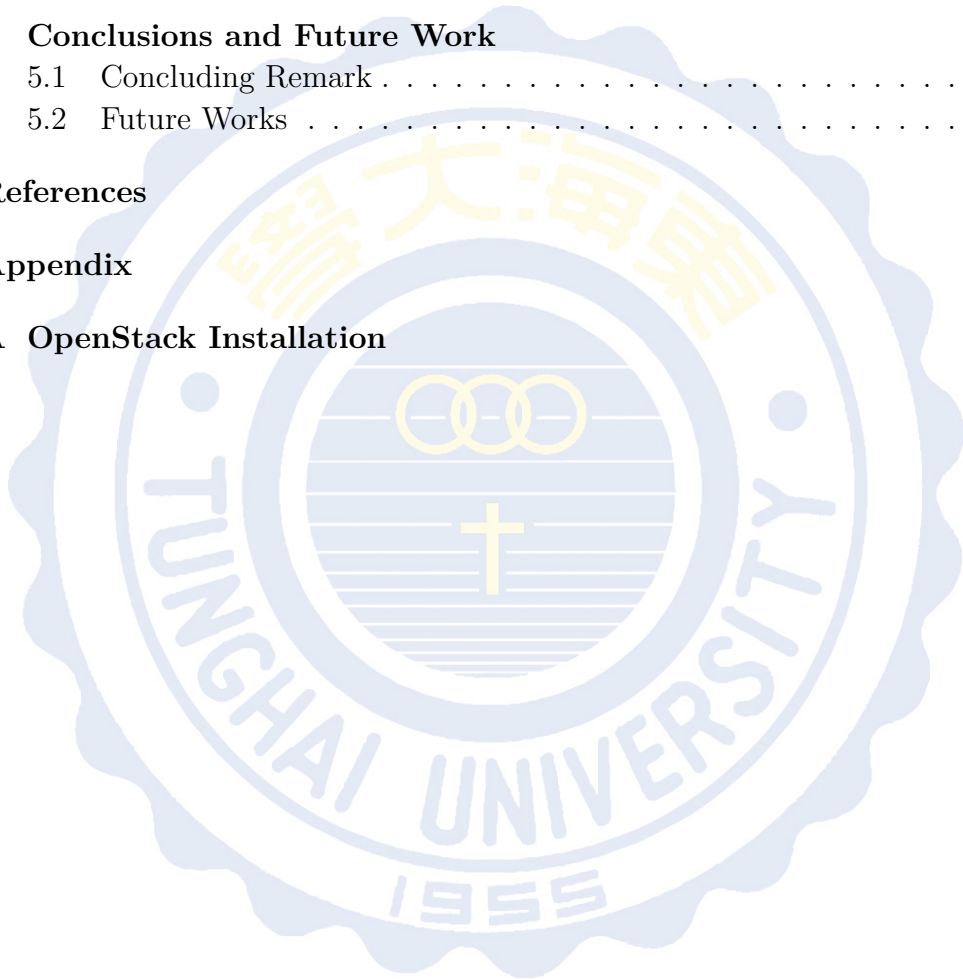
最後我要感謝我的家人，在這兩年來無論我遇到任何挑戰。都一直支持鼓勵我，有你們大家的支持，才能讓我沒有後顧之憂的完成學業。謝謝所有關心我的人，感謝。

東海大學資訊工程學系 高效能實驗室 李政岳 二零六年七月

Table of Contents

摘要	i
Abstract	ii
致謝詞	iv
Table of Contents	v
List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Goal and Contributions	2
1.3 Limitations of Prior Art	2
1.4 Thesis Organization	3
2 Background Review and Related Work	4
2.1 Background Review	4
2.1.1 Cloud Computing	4
2.1.2 Virtualization	6
2.1.3 OpenStack	8
2.1.4 OpenStack Component	8
2.1.5 OpenStack Conceptual Architecture	10
2.1.6 Graphics Processing Unit	12
2.1.7 Virtual Desktop Infrastructure	13
2.2 Related Works	17
3 System Design and Implementation	18
3.1 System Design Architecture	18
3.2 System Implementation	19
3.2.1 OpenStack Service Deployment	20
3.2.2 GPU Pass-Through	24
3.2.3 Methodology	24

4	Experimental Results	27
4.1	Experimental Environment	27
4.2	Network Delay Experiment	29
4.3	Heaven Benchmark Experiment	29
4.4	CineBench Experiment	41
5	Conclusions and Future Work	43
5.1	Concluding Remark	43
5.2	Future Works	44
	References	46
	Appendix	49
A	OpenStack Installation	49



List of Figures

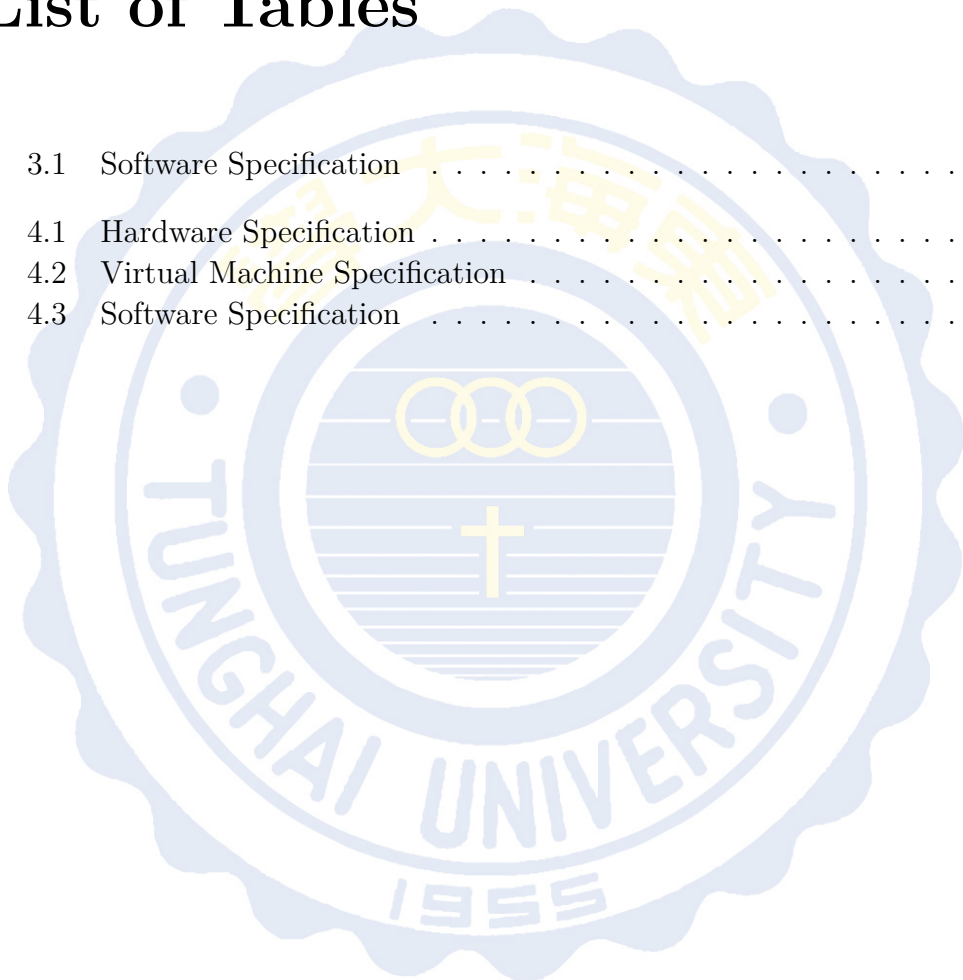
2.1	OpenStack Architecture	9
2.2	OpenStack conceptual Architecture	10
2.3	VM spawning sequence OpenStack	12
2.4	noVNC Sequence	16
2.5	Spice Architecture	16
3.1	System Architecture	19
3.2	OpenStack nova with KVM and XenServer	20
3.3	Experiment Environment	20
3.4	OpenStack Overview	21
3.5	OpenStack VM Instances	22
3.6	OpenStack noVNC Web Console	22
3.7	OpenStack Spice web console	23
3.8	OpenStack Spice client	24
3.9	GPU Pass-Through Architecture	25
3.10	NVIDIA GRID K2 on compute	25
3.11	NVIDIA GRID K2 on Windows VM	26
4.1	ping 60s	29
4.2	ping 60s	29
4.3	OpenStack VM HeavenBenchmark FPS	30
4.4	OpenStack VM HeavenBenchmark MaxFPS	30
4.5	OpenStack VM HeavenBenchmark MinFPS	31
4.6	OpenStack VM HeavenBenchmark Score	31
4.7	OpenStack and XenServer instance performance compare(4 vcpu)	32
4.8	OpenStack and XenServer instance performance compare(4 vcpu)	32
4.9	OpenStack and XenServer instance performance compare(4 vcpu)	33
4.10	OpenStack and XenServer instance performance compare(4 vcpu)	34
4.11	OpenStack and XenServer instance performance compare(8 vcpu)	34
4.12	OpenStack and XenServer instance performance compare(8 vcpu)	35
4.13	OpenStack and XenServer instance performance compare(8 vcpu)	35
4.14	OpenStack and XenServer instance performance compare(8 vcpu)	36
4.15	OpenStack and XenServer instance performance compare(16 vcpu)	36
4.16	OpenStack and XenServer instance performance compare(16 vcpu)	37
4.17	OpenStack and XenServer instance performance compare(16 vcpu)	37
4.18	OpenStack and XenServer instance performance compare(16 vcpu)	38

4.19 Compare Physical FPS 38
4.20 Compare Physical Max FPS 39
4.21 Compare Physical Min FPS 39
4.22 Compare Physical score 40
4.23 CineBench CPU 41
4.24 CineBench OpenGL 41



List of Tables

3.1	Software Specification	21
4.1	Hardware Specification	27
4.2	Virtual Machine Specification	28
4.3	Software Specification	28



Chapter 1

Introduction

In the current era of cloud computing, computing, storage, network resources are virtualized, the data center resources in accordance with the needs of users to allocate. OpenStack [1–3] is now one of the most widely used open source cloud platform, in recent years is very popular, many of which are interesting questions worthy of our discussion.

1.1 Motivation

In today's cloud computing era [4–6], computing, storage, network resources are virtualized, the data center resources in accordance with the needs of the user distribution is the current trend. OpenStack is now one of the most widely used open source cloud platform, in recent years is very popular, many of which are interesting questions worthy of our discussion. Today's cloud services are not limited to IaaS, PaaS, SaaS [7–11] model has emerged more and more different service model. Today's global companies have cited cloud technology, IaaS providers, including IBM, HP, Microsoft, Google, Amazon, Rackspace, and several other companies, to provide services to small individual users or other large non-information type of the company. So that these enterprises can not build their own cloud environment, focus on their own business, and can also use the convenience of the

cloud. In addition to the convenience of the cloud, how to provide users with an easy to use the use of the environment, so that users have a better user experience, is a very important issue now. The closest part to the average user is the virtual desktop, the virtual virtual desktop can be configured through virtualization resources to configure the operating system in accordance with the needs of users, and can be installed on the operating system of different applications. But the virtual desktop is limited to the network or other reasons, resulting in the use of non-smooth situation will occur, so this paper plans to use PCI Pass-through technology to use GPU resources to accelerate the performance of virtual desktops, enhance the virtual desktop The degree of smoothness.

1.2 Thesis Goal and Contributions

The current OpenStack-supported hypervisors include KVM, QEMU, VMware VSphere, XenServer, Hyper-V, and so on. Different hypervisors will have their own VDI. There are generally noVNC, Spice, RDP, etc., and users will use VDI Connect to the hypervisor on the hypervisor, but not all VDI can support the GPU display, so this article will be OpenStack integrated XenServer as a hypervisor layer, and through the Pass-through technology GPU (this article using NVIDIA Grid K2) configuration to Virtual machines, and the use of the RDP (Remote Desktop Protocol) environment to achieve the ability to accelerate virtual machine picture processing, and through different benchmarking software to test performance, and to assess the GPU and vCPU virtual desktop performance impact.

1.3 Limitations of Prior Art

The data center network will be affected too much, including the workload over the load caused by the network congestion, as well as cable failure, set the error, etc., although the CPU Benchmark and GPU Benchmark currently have a lot of software available to us Do a test, and in the cloud environment to do so

will have some software compatibility issues. In addition, this article describes the OpenStack integrated GPU part, the current OpenStack does not provide full GPU virtualization solution, so now only through the pass-through way to provide the GPU to the virtual machine. Also in the virtual desktop part of the existing VDI, not all support GPU display part, some VDI will lead to virtual desktop can not be used normally, only a specific hypervisor with a specific VDI can be used normally.

1.4 Thesis Organization

In the second section, we will introduce some background knowledge, including cloud computing, OpenStack, Neutron, GPU, VDI, and in the third part of the section we will introduce the experimental environment, as well as experimental methods and steps. The fourth part of the chapter will introduce and analyze the results of the experiment, the final part of the end, we will make a conclusion, and sort out our contribution and future prospects of the part.

Chapter 2

Background Review and Related Work

In this chapter, we review some background knowledges for later use of system design and implementation.

2.1 Background Review

2.1.1 Cloud Computing

Cloud computing is a web-based computing model that provides a wide range of resources, including operations, storage, and so on, that typically provide resources in a large number of resources in the data center through virtualized and dynamically expanded Do not need to understand the infrastructure of the cloud infrastructure or do not need to directly master the cloud technology, but it can also use the resources of the cloud.

And according to the definition of NIST basically by the three service models, four deployment models, five features to form.

The three service modes are as follows

- Software as a Service (SaaS): The user uses the software that has been deployed in the cloud or has been stored in the cloud, and does not use the cloud infrastructure and the cloud environment's programming environment. In this way, users do not need to install their own software on their own computers, thereby reducing maintenance difficulties, including software compatibility issues.
- Platform as a Service (PaaS): PaaS provides a computing platform that includes the operation of the cloud platform, the database, the programming environment, and the Web server. The application developers can develop and implement the software on PaaS without having to spend a lot of money to procure the physical machines and manage them maintain.
- Infrastructure as a Service (IaaS): IaaS is the most basic kind of cloud service model, providing physical machines, virtual machines and other resources, so that users can control the allocation of resources. But can not control the cloud's infrastructure.

The four deployment modes are as follows

- Public Cloud: Through the public network so that users can use, but can not see other people's information.
- Private Cloud: Private cloud compared to the public cloud, some aspects of more advantages, EX: flexibility, to provide appropriate services, and the public cloud is the difference between the private cloud program and information are usually business or user management.
- Community Cloud: Community clouds shared by several organizations are often used by specific communities that share common needs.
- Hybrid Cloud: The cloud may contain two or more deployment patterns.

Five features are as follows

- On-demand self-service.
- Anytime, anywhere access by any network device.
- Resource pooling.
- Quick redeployment.
- Can be monitored and measured.

2.1.2 Virtualization

In the cloud computing technology, virtualization [12–15] is the computer entity resources such as servers, networks, memory and storage to be abstract, converted after the show, so that users can be better than the original way to apply these resources. The virtual part of these resources is not limited by the way in which existing resources are erected, geographically or physically. Generally refers to the virtualization resources, including computing power and data storage.

In computer science, virtualization is a technology for a computer or operating system. Virtualization hides the real hardware device for the user, presenting another virtual computing platform. Virtualization technology transforms the entity into a virtual computing environment (virtual machine) to the user. And the user uses the client's application to operate the virtual machine, virtual machine does not limit any application or operating system, the virtual machine is like running directly on the same machine. Virtual machines are unified management of hardware resources (such as networks, screens, keyboards, hard drives) at a more restrictive level than a processor or memory, the client will be restricted to access the peripherals of the entity, This depends on the access strategy adopted by the entity.

Virtual Desktop (VDI) is a software technology that separates the desktop environment and related applications from the entity's client devices. Virtual desktops can be combined with application virtualization and user profile management systems, now known as "user virtualization", providing a comprehensive

virtual desktop environment management system. In this mode, the components required by the desktop are virtualized. This gives virtual desktops more flexibility and a more secure environment. In addition, this approach supports a variety of disaster recovery strategies, because all the components are basically stored in the data center, and through the maintenance system backup. If the user's components or files are lost, but also easy to restore, because basically all of the desktop components can be registered in other entities on the machine. In addition, since the data is not stored in the user's equipment, if the user's device is lost, it can also reduce the loss, the data are stored in the data center. The following is a more detailed description of the type of desktop virtualization technology that will be used in a typical deployment.

The implementation of desktop virtualization is categorized if the virtual desktops are running remotely or locally, whether the access requirements are constant or designed to be intermittent, and the virtual desktops still exist between communications. Often, software products that provide desktop virtualization solutions can be combined with local and remote implementations to provide specific and appropriate requirements. The client device's independent functionality is interdependent with the server's location and access policies. Virtualization is used to render independent desktops to multiple user users.

Build your virtualization and simplify your IT infrastructure with virtualization solutions. Virtualization can help reduce capital expenditures through server consolidation, use automation to reduce operating costs, and reduce scheduled and non-scheduled downtime, significantly reducing revenue losses. Reduce capital and operating costs by improving energy efficiency while leveraging server consolidation to reduce the need for hardware.

2.1.3 OpenStack

In the related technologies of cloud service, virtualization technology plays a decisive role and distributed many kinds of virtualization project. Through the beginning of VMware vSphere and Hyper-V deployment until using OpenStack [16, 17] platform.

OpenStack is a NASA open source software jointly developed by NASA and Rackspace, licensed as an Apache license, and is a free software and open source project to build Infrastructure as a Service.

OpenStack has three module modules, Netcom modules and storage modules, plus a centralized management of the dashboard module, to form a set of OpenStack shared services, and to provide virtual machine, external operations Resources to facilitate flexibility expansion or scheduling. Users can use open source OpenStack to build their own Amazon EC2-like services, and OpenStack specifications are also compatible with Amazon EC2, so whether it is in the above development system, the use of systems, or to help people develop the system, people use the system , OpenStack can be achieved, which is now the reason for such open source IaaS hot.

2.1.4 OpenStack Component

OpenStack mainly has seven different functions of the suite, namely the computing suite Nova, the object storage kit Swift, block storage kit Cinder, Netcom kit Neutron, identity identification kit Keystone, image management suite Glance, dashboard suite Horizon.

- Nova: Provides the ability to deploy and manage virtual machines.
- Swift: A decentralized storage platform that can store unstructured data.
- Neutron: To ensure the consistency and reliability of the entire network.

- Cinder: Provide block storage capacity, with snapshot function.
- Keystone: Provides a variety of authentication methods to see which users can access which services.
- Glance: Provide image search, registration and service delivery and other functions.
- Horizon: Graphical web interface allows IT staff to manage the hardware resources of the cloud service.

The OpenStack architecture is shown in Figure 2.1.

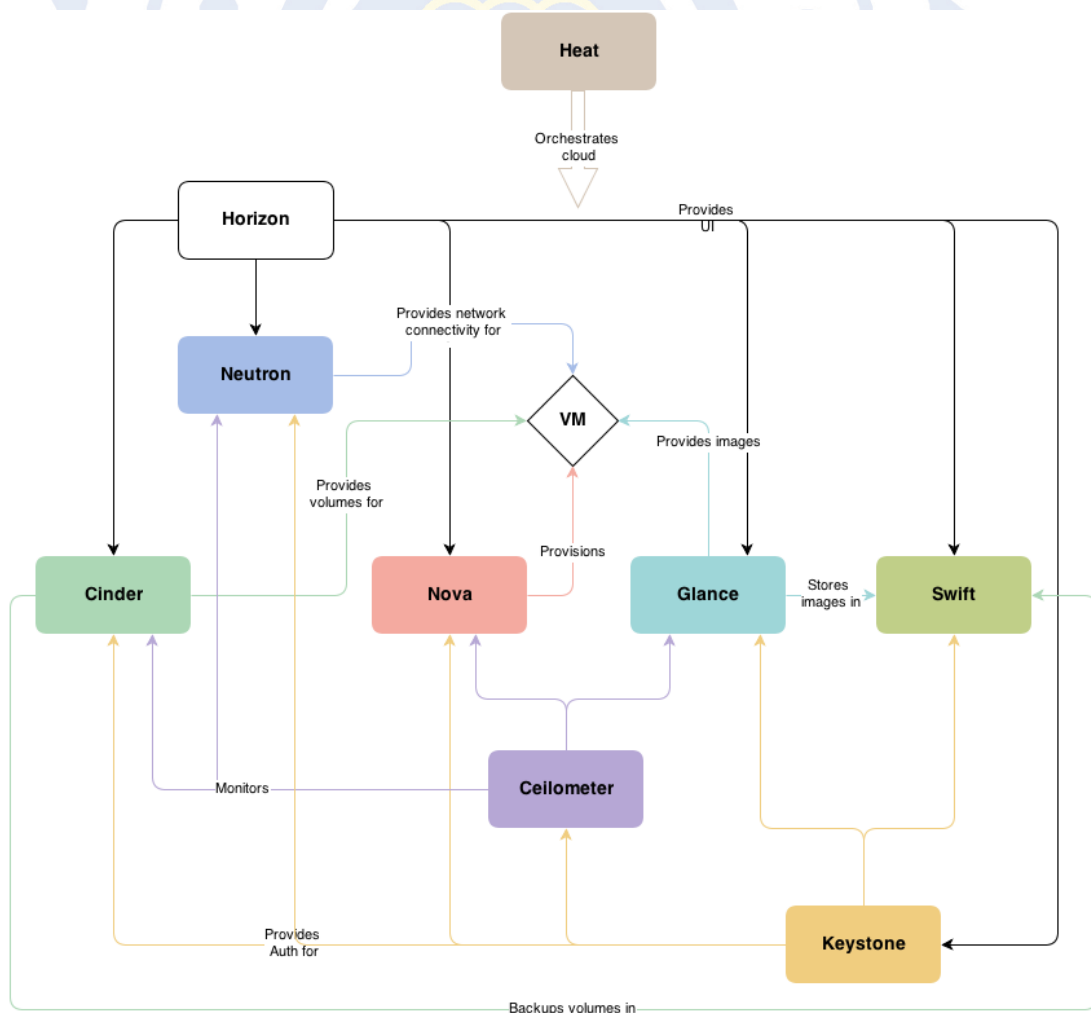


FIGURE 2.1: OpenStack Architecture

2.1.5 OpenStack Conceptual Architecture

The OpenStack Conceptual Architecture is shown in Figure 2.2. Virtual machine generation is one of the most important use-cases in any cloud environment. Here we describe the steps involved in configuring the instance in the OpenStack cloud, which includes the order of the requests and the interaction between the various OpenStack components in order to successfully start the VM. Here we describe the steps involved in configuring the instance in the OpenStack cloud, which includes the order of the requests and the interaction between the various OpenStack components in order to successfully start the VM.

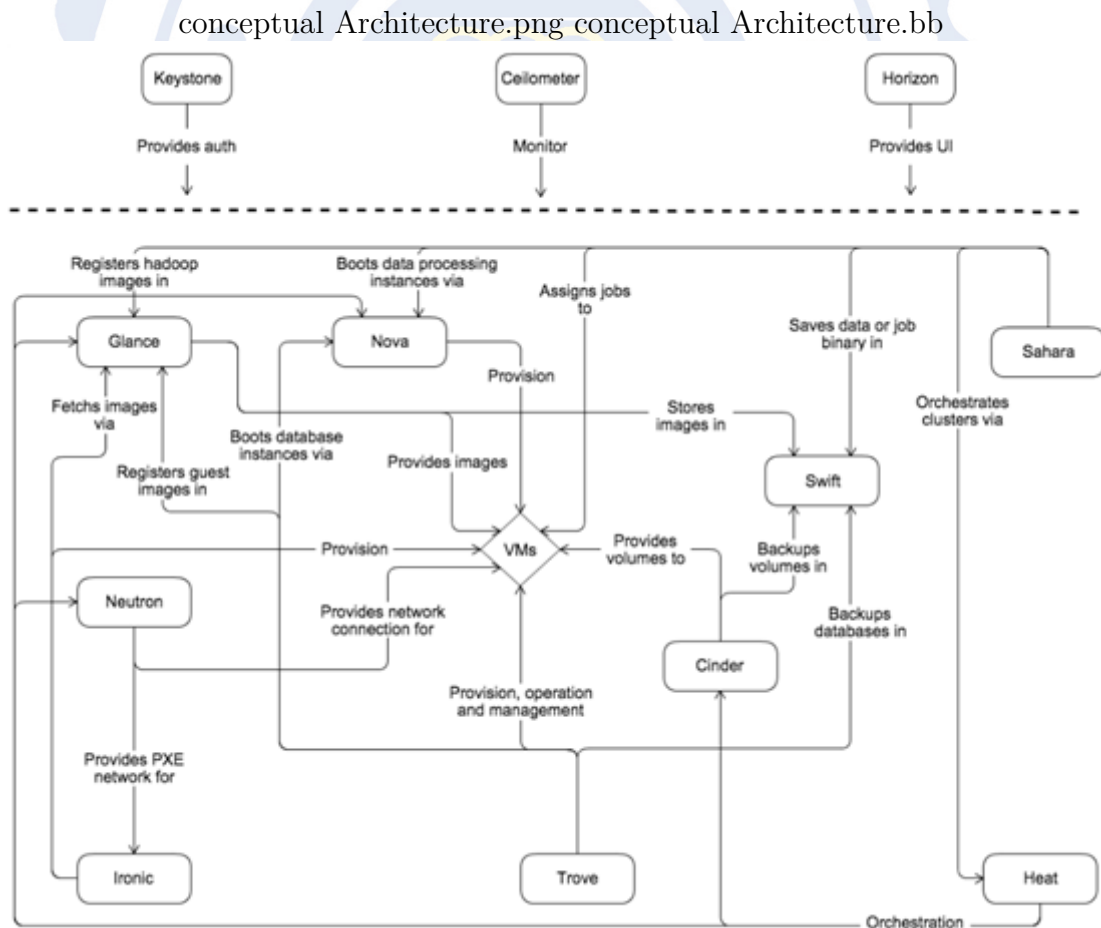


FIGURE 2.2: OpenStack conceptual Architecture

And Figure 2.3. is the OpenStack suite of communication process. These connections are initiated by using the associated API as a remote program call (RPCs) that can be converted to an nova-boot command when the tenant issues

an instance request via the CLI (command line interface) or dashboard. The Nova API server sends the user's credentials to Keystone for authentication (1, 2). After successful authentication, nova-api contacts nova-db to initialize the initial configuration information for the new instance into the database (4,5,6,7). Then, nova-api sends an RPC call to the nova-scheduler, requesting the ID of the hosts that started the instance (8, 9). The nova-scheduler crawls information to nova-db and uses the filter and weighting functions to select the best (or least load) HOST and return its ID (10, 11, 12). The scheduler selects the appropriate compute node as the host and sends a message to start the new instance (12, 13). Nova-compute then RPC calls to nova-conductor, nova-conductor access nova-compute to nova db for information such as host ID, flavor disk, and vCPU (14, 15, 16, 17, 18). Use the authentication token, nova-compute to make a REST call to the glance-api to retrieve the image from the image repository and upload it to the selected host (19, 20, 21). This uploaded image will be cached for future use. Subsequently, nova-compute calls neutron-api to retrieve network allocation and configuration information so that fixed IP is assigned to the new instance (22, 23, 24). If the user requests to attach some volumes to the instance, nova-compute uses the REST call, the additional volume (25, 26, 27) for the cinder-api. Finally, nova-compute forwards all the information to the virtualization driver and generates an instance request on hypervisor (28). During the various phases of the configuration process, the corresponding instance states that can be seen from the Horizon dashboard are: scheduling> Networking> Spawning> Running.

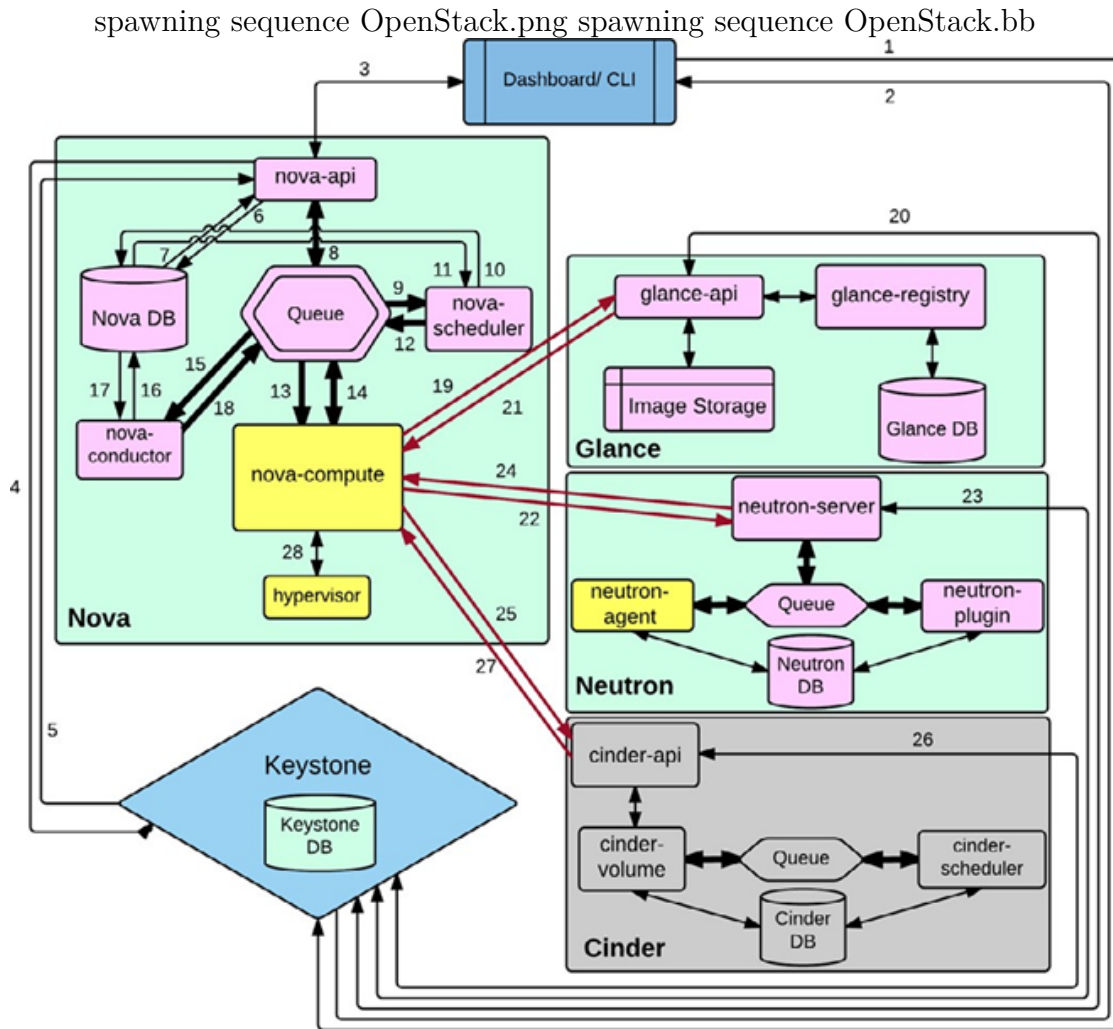


FIGURE 2.3: VM spawning sequence OpenStack

2.1.6 Graphics Processing Unit

Graphics processing unit (GPU), also known as the display core, visual processor, display chip, is a specialized in personal computers, workstations, game consoles and mobile devices (such as tablet PCs, smart phones, etc.) The work of the microprocessor is treated as the heart of the card. In recent years, experts from various fields have noticed the computing power of the GPU and have tried to apply its computing power extensively to data analysis (eg, machine learning, depth learning, etc.), and huge amounts of data processing (eg MapReduce). GRID vGPU is NVIDIA's graphics acceleration technology, which can use a single GPU to achieve multiple virtual desktop services GPU sharing, through the NVIDIA

GRID GPU card installed on the X86 host, a substantial increase in the VMware vSphere. The performance of running a graphics-intensive application is a great help for users who need to use a large number of 2D and 3D graphical interfaces, such as architects, engineering labs, and clinicians in medical facilities. NVIDIA GRID vGPU's technology offers the following better user experience:

- For all graphics applications with 100 per cent compatibility, access to each PC and workstation NVIDIA graphics card driver, can provide to each virtual machine and Local side of the same performance.
- Support real-time integration, unified centralized management of graphics data set resources, and then by the end user needs to provide resources to meet the needs.
- Provide a decentralized workforce productivity in different workspaces (such as animation designers, clinicians and researchers).
- With NVIDIA's high-end graphics processing performance, server managers can deliver better performance services without compromising the original vSphere desktop virtualization environment.
- Reduce costs and increase resource utilization by sharing the GPU.
- Centralized management of graphic data sets to provide better protection for data storage.

2.1.7 Virtual Desktop Infrastructure

To understand desktop virtualization technology, you must understand the development of desktop virtualization. You can simply desktop virtualization [18–21] technology is divided into the following stages: Large host era: the mainframe when the first appearance of the price is very expensive, although expensive but the mainframe computing power is very good, so it was proposed to share a machine to multiple users to use the way, of course, this is not virtualization, But

rely on the system of multiple users of the multi-task form. For example, Linux, Unix and Windows server versions can support multi-user form. Desktop virtualization agreement: Remote Desktop Protocol (RDP, Remote Desktop Protocol) early from Microsoft from Citrix company bought the technology developed by itself.

And independent computing structure agreement (ICA, Independent Computing Architecture) is Citrix company to use So far the service. The first phase of desktop virtualization technology actually combines remote desktop connectivity and virtual systems, allowing users to have their own virtual desktop system. Can do so of course because the server's hardware has been from the previous single-core single-work into a multi-core multi-work, to enhance the server's computing power and virtualization capabilities, and memory from the previous 4GB, 8GB breakthrough to 128GB Of the capacity, greatly enhance the server's computing power, enhanced hardware capabilities coupled with the maturity of virtualization, enabling servers to provide multiple virtual desktops to provide user operation, making desktop virtualization technology to become a large-scale application may. The main vendors that currently offer desktop virtualization solutions are Microsoft, VMware, and Citrix, which uses three different protocols. Microsoft's RDP (Remote Desktop Protocol) was developed by Citrix and was later developed by Citrix Microsoft has purchased and improved the agreement in Windows Server, and VMware Horizon View also supports such agreements; Citrix will apply the agreement to its virtualization products; the third is the US company Teradici developed PCoIP (PC-over-IP) agreement, was later developed by the Citrix self-developed ICA (Independent Computing Architecture) VMware is purchased and applied to its desktop virtualization products to enhance the user's desktop virtualization experience. Virtual desktop system attaches great importance to the agreement, the agreement often determines the user experience is good or bad. It is known from the original technical documents that the original ICA (Independent Computing Architecture) protocol is superior to the RDP (Remote Desktop Protocol) and PCoIP (PC-over-IP) protocols, requiring about 30kbps of bandwidth, while the RDP (Remote Desktop Protocol) In 50kbps or so, this does not include

playing games and watching movies and 3D graphics under the state of the loss of bandwidth, just the transmission of the screen there is such a consumption.

RDP (Remote Desktop Protocol) agreement currently causes the most bandwidth loss, will cause the impact is running in the WAN environment, play video, Flash, the implementation of 3D software, such as screen delay and distortion will occur. ICA (Independent Computing Architecture) agreement user experience will be very smooth. VMware will improve the performance of the PCoIP agreement to enhance the virtual desktop user experience and published in the latest VMware View 5.0 products, according to the official file will reduce the bandwidth loss rate of 75 percent, is also leading all the virtual desktop agreement. These three vendors have their own virtualization server technology, Microsoft HyperV, VMware for vSphere, Citrix is XenServer, but can be installed on Hyper-v and vSphere.

This paper mainly discusses two kinds of remote connection technologies commonly used in OpenStack, namely noVNC and SPICE.

- noVNC: To provide a remote console or remote desktop access to guest virtual machines, use VNC or SPICE HTML5 through either the OpenStack dashboard or the command line. Both client proxies leverage a shared service to manage token authentication called nova-consoleauth. This service must be running for either proxy to work. Many proxies of either type can be run against a single nova-consoleauth service in a cluster configuration. Do not confuse the nova-consoleauth shared service with nova-console, which is a XenAPI-specific service that most recent VNC proxy architectures do not use.

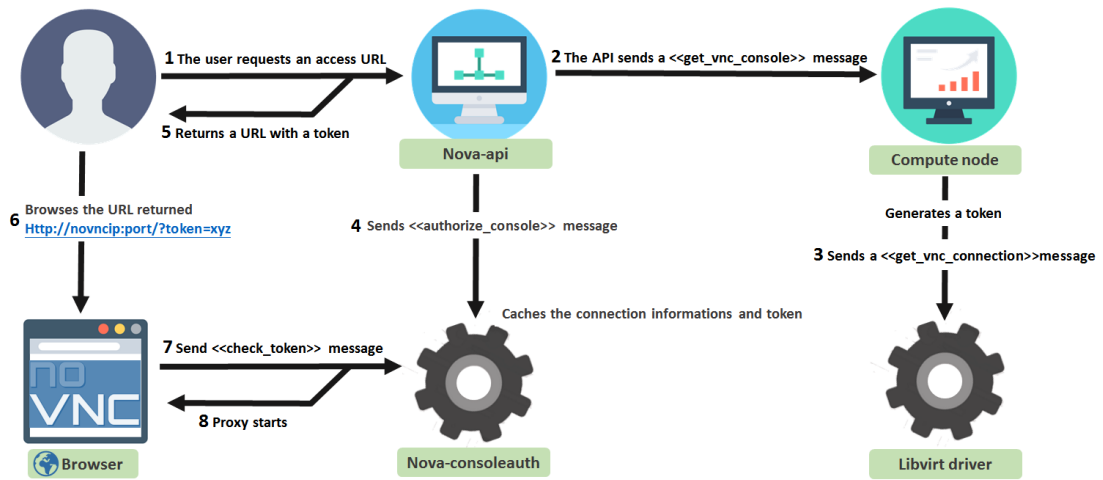


FIGURE 2.4: noVNC Sequence

- Spice: OpenStack Compute supports VNC consoles to guests. The VNC protocol is fairly limited, lacking support for multiple monitors, bi-directional audio, reliable cut-and-paste, video streaming and more. SPICE is a new protocol that aims to address the limitations in VNC and provide good remote desktop support. SPICE support in OpenStack Compute shares a similar architecture to the VNC implementation. The OpenStack dashboard uses a SPICE-HTML5 widget in its console tab that communicates to the nova-spicehtml5proxy service by using SPICE-over-websockets. The nova-spicehtml5proxy service communicates directly with the hypervisor process by using SPICE.

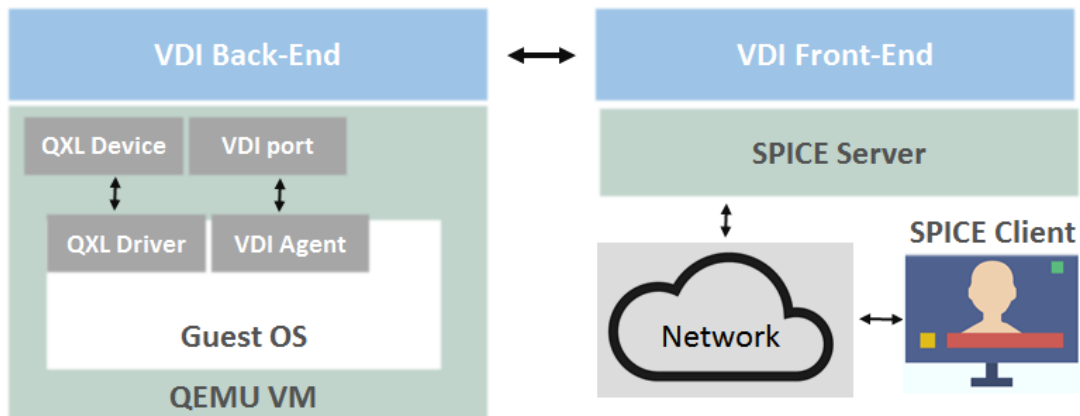


FIGURE 2.5: Spice Architecture

2.2 Related Works

Several recent studies proposed broad guidelines for the development of benchmarks of cloud resources, among them are the works of Folkerts et al., O' Loughlin et al., Binnig et al. and Rak et al. There have also been several benchmarking studies on actual cloud deployments. Mailk et al. [22] In their measurement study evaluate OpenStack's open source SDN layers, using Neutron together with a OpenDaylight, OFAgent, ML2 and Ryu used in OpenStack Juno in the face of different types and severity levels of network errors. Pacevič et al. [23] Their paper presents the development of visualization software as a service in the OpenStack cloud infrastructure. VisLT cloud visualization service is developed for visualizing the results computed and stored in the private cloud infrastructure. GPU is virtualized as a PCI device employing direct pass-through [24, 25] technology on the hardware virtual machines of Xen hypervisor to ensure fast remote rendering, which is a key feature of distributed visualization systems. Iserte et al. Their study the viability of this approach using a public cloud service configuration, and develop a module for OpenStack in order to add support for the virtualized devices and the logic to manage them. The results demonstrate this is a viable configuration which adds flexibility to current and well-known cloud solutions. Yamato et al. [26, 27] Their study proposes a PaaS which analyzes application logics and offloads computations to GPU and FPGA automatically when users deploy applications to clouds.

Chapter 3

System Design and Implementation

Because the popularity of the cloud, and now more and more people will come into contact with the relevant environment, most people most commonly used part of the virtual desktop, and the general virtual desktop performance is low, resulting in operational difficulties. The work is to integrate the GPU on OpenStack , and through the Pass-through way to allow the virtual machine can use the GPU resources, to speed up the VDI performance. In this section will introduce our overall architecture design, as well as the use of open source software.

3.1 System Design Architecture

As shown in Figure 3.1 , We used OpenStack as our foundation and used as the basis for the entire virtualization. We integrated the GPU on OpenStack and passed the Pass-through way to allow the virtual machine to use the GPU's resources and test it in a number of VDI to find a VDI protocol for OpenStack.

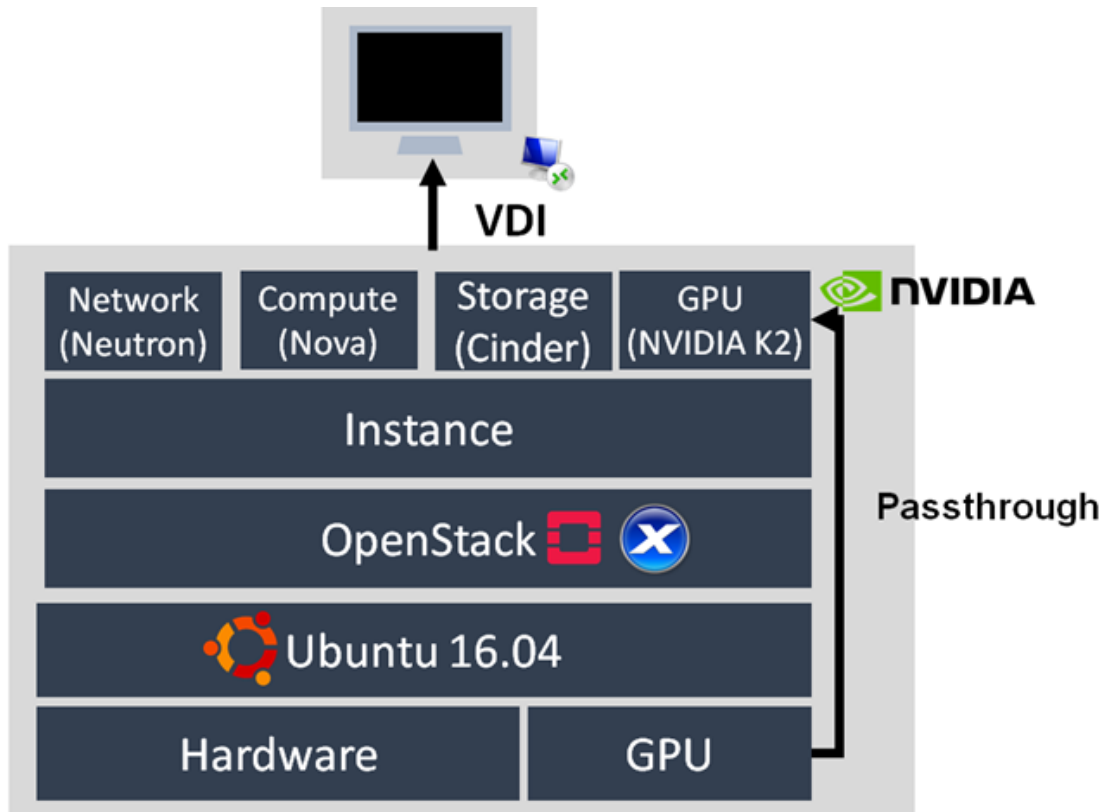


FIGURE 3.1: System Architecture

3.2 System Implementation

The proposed architecture is divided into several parts to do research, one for the OpenStack virtual machine performance test. Again, GPU performance testing and performance comparison in different network environments.

In this work, we built an OpenStack cluster of eight physical machines, one for the controller Four of which are compute nodes and equipped with K2 GPU, compute node, respectively, there are different hypervisors, KVM, XenServer, ESXI, etc., this node using XenServer as the end of the experiment, the other two nodes were Network node and the Storage node, Network Node is responsible for the transfer of network packets and vlan management, and in the storage part we use Cinder and Swift suite, Cinder is Block Storage service, in this paper we will Cinder split out of space , Used as a virtual machine hard drive, and Swift is the Object service, where we used as a storage file for XenServer.

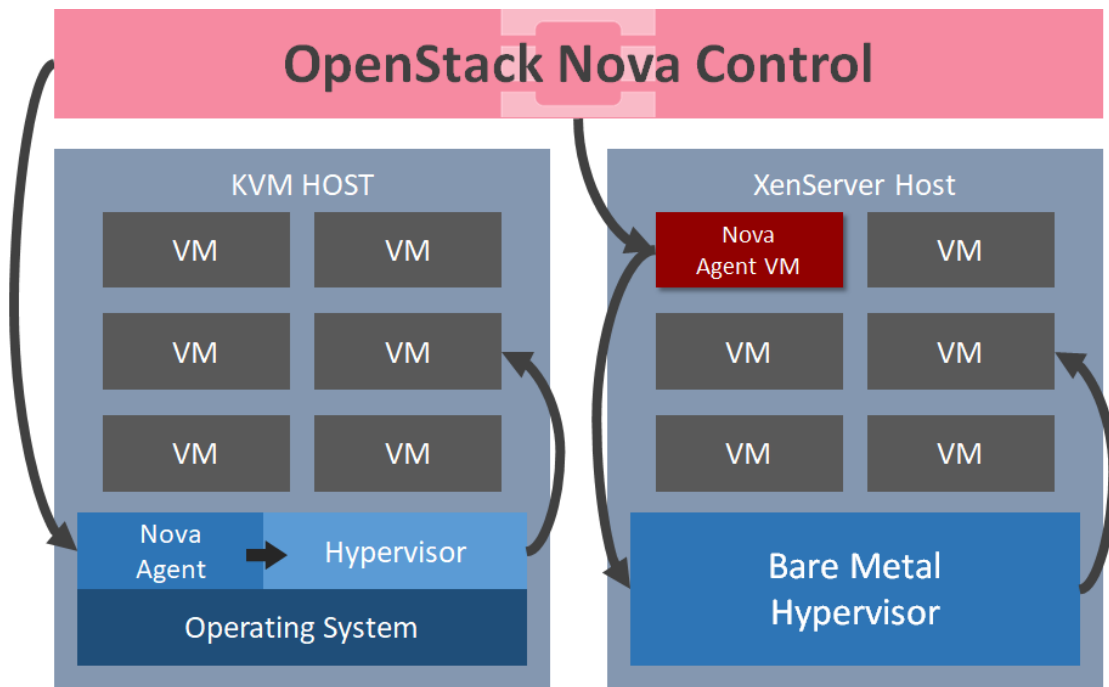


FIGURE 3.2: OpenStack nova with KVM and XenServer

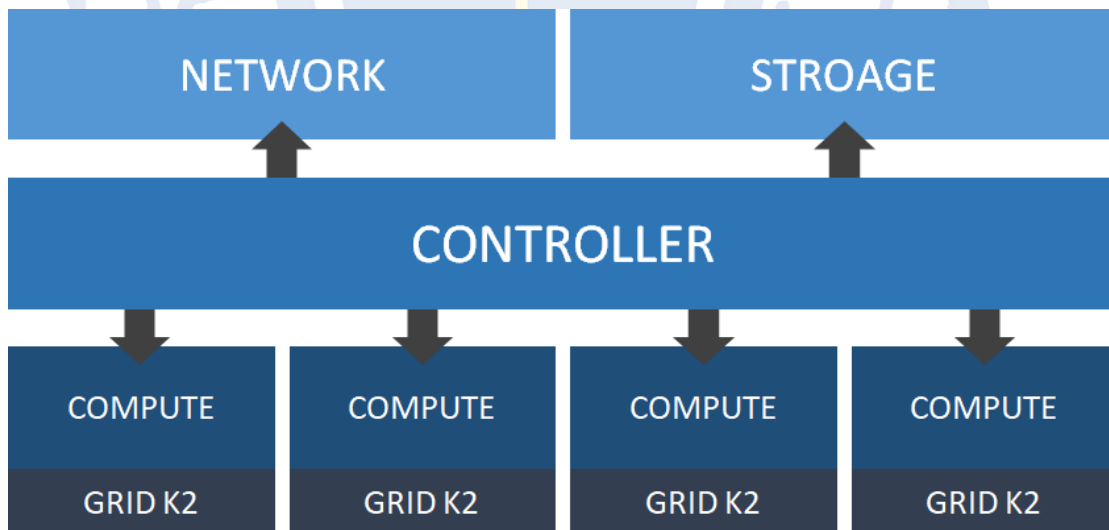


FIGURE 3.3: Experiment Environment

3.2.1 OpenStack Service Deployment

By using Ubuntu OS to create virtual machines, open source software OpenStack is applied to build and manage the proposed cloud system. The overview of the system is shown in Figure 3.4 and Figure 3.5.

TABLE 3.1: Software Specification

Software	Version
Ubuntu	16.04.02 LTS
Windows	10
OpenStack	Newton
Python	2.7.6
MariaDB	10.1.1.14
XenServer	7.0
NVIDIA Driver	369.95

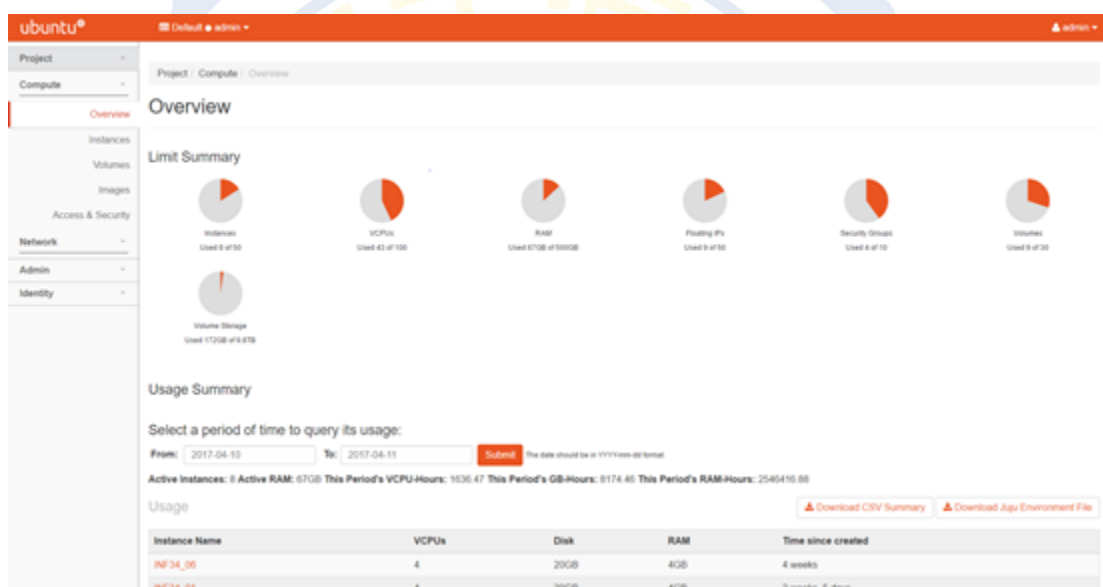


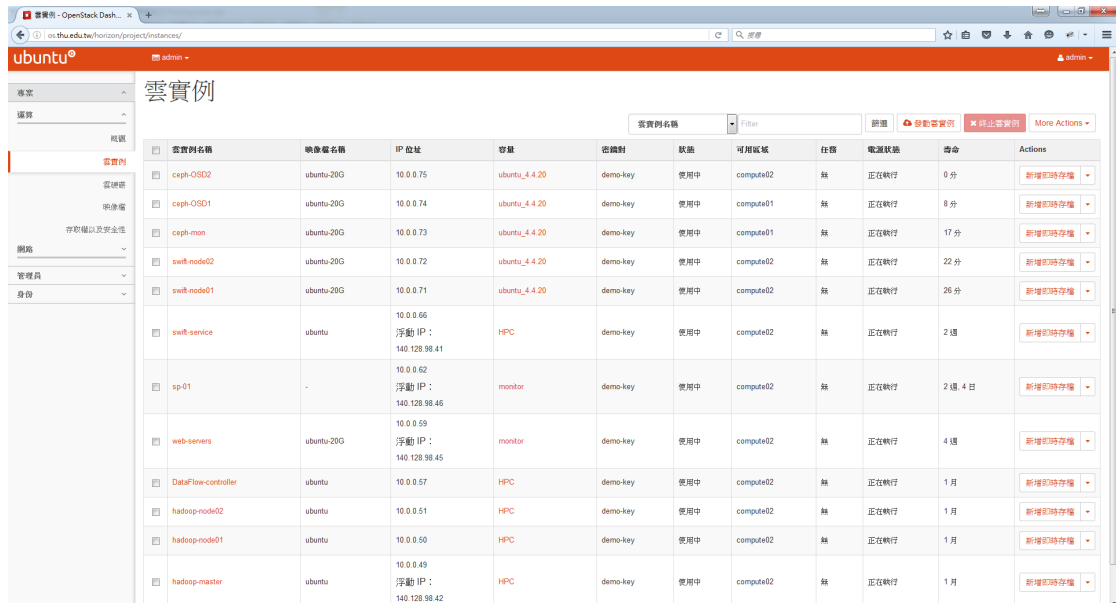
FIGURE 3.4: OpenStack Overview

VDI Deployment noVNC

OpenStack uses noVNC to implement user interface to virtual desktop, and noVNC is a Web client that uses Web Scket and HTML5 Canvas. But the use of noVNC will cause the use of virtual desktop delay and Lag status, for the user is very inconvenient to use.

VDI Deployment Spice Protocol

Because noVNC is not smooth, we have integrated Spice (The Simple Protocol for Independent Computing Environments) services in our OpenStack cloud platform, SPICE allows users to view the "desktop" environment. The Spice environment in our system is shown in Figure 3.7 and Figure 3.8.



The screenshot shows the OpenStack VM Instances dashboard. The title is '雲實例' (Cloud Instances). The table lists various instances with columns for Name, Image Name, IP Address, Flavor, Key Pair, Status, Availability Zone, Task State, Power State, and Age. The instances include ceph-OSD2, ceph-OSD1, ceph-mon, swift-node02, swift-node01, swift-service, sp-01, web-servers, DataFlow-controller, hadoop-node02, hadoop-node01, and hadoop-master.

雲實例名稱	映像名稱	IP 位址	容量	密碼對	狀態	可用區域	任務	電源狀態	壽命	Actions
ceph-OSD2	ubuntu-20G	10.0.0.75	ubuntu_4.4.20	demo-key	使用中	compute02	無	正在執行	0分	新增即時存檔
ceph-OSD1	ubuntu-20G	10.0.0.74	ubuntu_4.4.20	demo-key	使用中	compute01	無	正在執行	9分	新增即時存檔
ceph-mon	ubuntu-20G	10.0.0.73	ubuntu_4.4.20	demo-key	使用中	compute01	無	正在執行	17分	新增即時存檔
swift-node02	ubuntu-20G	10.0.0.72	ubuntu_4.4.20	demo-key	使用中	compute02	無	正在執行	22分	新增即時存檔
swift-node01	ubuntu-20G	10.0.0.71	ubuntu_4.4.20	demo-key	使用中	compute02	無	正在執行	26分	新增即時存檔
swift-service	ubuntu	10.0.0.66 浮動 IP : 140.128.98.41	HPC	demo-key	使用中	compute02	無	正在執行	2週	新增即時存檔
sp-01	-	10.0.0.62 浮動 IP : 140.128.98.46	monitor	demo-key	使用中	compute02	無	正在執行	2週, 4日	新增即時存檔
web-servers	ubuntu-20G	10.0.0.59 浮動 IP : 140.128.98.45	monitor	demo-key	使用中	compute02	無	正在執行	4週	新增即時存檔
DataFlow-controller	ubuntu	10.0.0.57	HPC	demo-key	使用中	compute02	無	正在執行	1月	新增即時存檔
hadoop-node02	ubuntu	10.0.0.51	HPC	demo-key	使用中	compute02	無	正在執行	1月	新增即時存檔
hadoop-node01	ubuntu	10.0.0.50	HPC	demo-key	使用中	compute02	無	正在執行	1月	新增即時存檔
hadoop-master	ubuntu	10.0.0.49 浮動 IP : 140.128.98.42	HPC	demo-key	使用中	compute02	無	正在執行	1月	新增即時存檔

FIGURE 3.5: OpenStack VM Instances

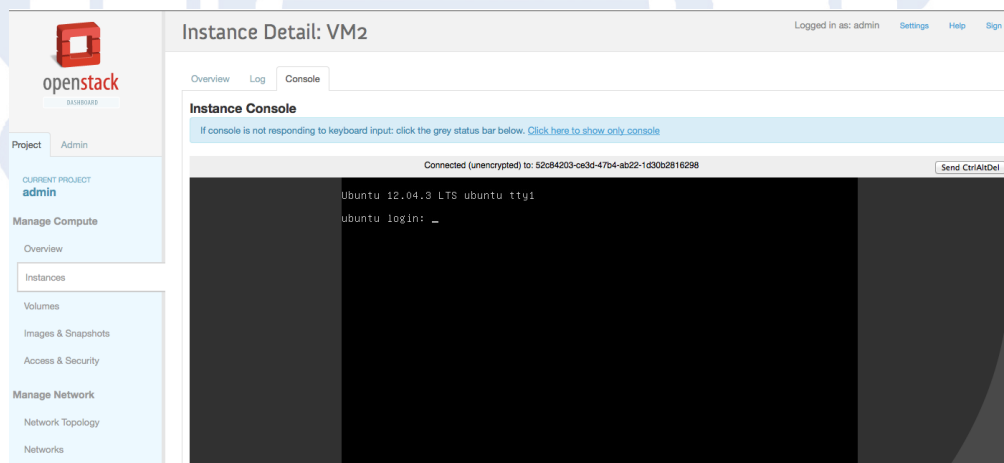


FIGURE 3.6: OpenStack noVNC Web Console

Graphic Processing Unit

This paper predicts the use of the GPU to speed up VDI fluency, as well as the ability of virtual machines to process graphics, a GPU that specializes in performing graphics operations on a PC, Server, or other device. In this section we will use NVIDIA Grid K2 as our GPU device, K2 is the use of NVIDIA Kepler architecture of the device, is designed to provide a virtual environment in the rich design experience designed.



FIGURE 3.7: OpenStack Spice web console

GRID GPU Features

- GPU Virtualization: NVIDIA KEPLER architecture is designed to provide virtual machine designed to provide a GPU hardware virtualization capabilities, the meaning of representatives can allow multiple users to use the GPU together, but the OpenStack KVM architecture does not support the underlying GPU virtualization capabilities.
- Low latency remote display: GRID with low-latency remote display technology, users can reduce the user through the VDI Protocol connection to the virtual machine when using the delay time, to improve the user experience, this technology will be directly to the virtual screen VDI Protocol.

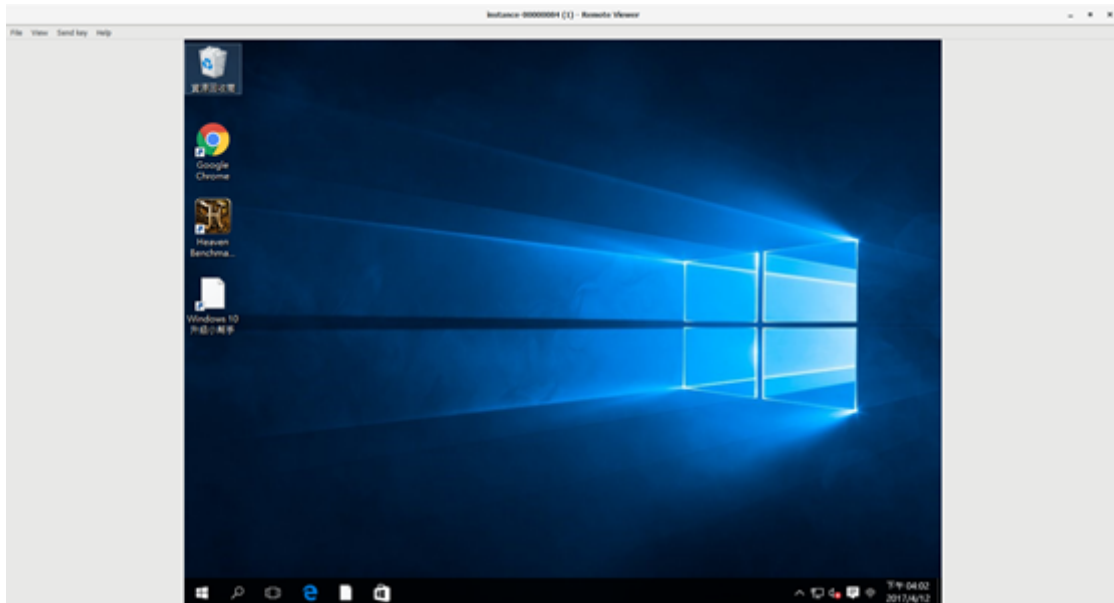


FIGURE 3.8: OpenStack Spice client

3.2.2 GPU Pass-Through

GPU resources can be configured through the GPU virtualization or Pass-Through way to configure the virtual machine, and Pass-Through practice, can be interpreted as a GPU dedicated to a virtual machine, through this technology, you can make specific Of the virtual machine configuration to all the GPU resources, with the vGPU is a different part of the vGPU is a complete GPU resources cut into smaller vGPU resources, provided to the need for virtual machines, and Pass-Through is complete Resource allocation to the same virtual machine.

3.2.3 Methodology

This article through the OpenStack and integrated XenServer, the program through Pass-through way will be installed on the compute node (XenServer) on the GPU (NVIDIA K2) assigned to the use of virtual machines, and designed several experiments, through the experiment Data to see if the GPU will affect the display performance of the virtual machine and test in different circumstances, including the memory size, vCPU core number, etc. will be associated with the GPU, trying

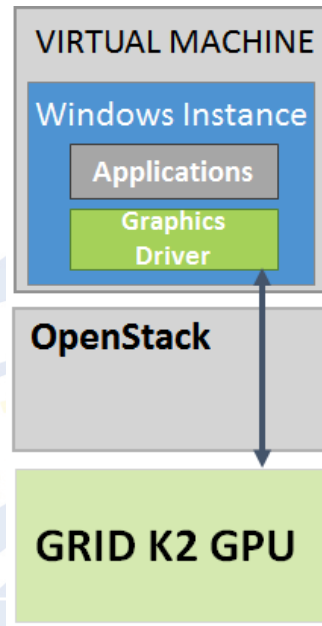


FIGURE 3.9: GPU Pass-Through Architecture

```

hpc@compute04:~$ lspci |grep NVIDIA
01:00.0 VGA compatible controller: NVIDIA Corporation GT218 [GeForce 210] (rev a2)
01:00.1 Audio device: NVIDIA Corporation High Definition Audio Controller (rev a1)
06:00.0 3D controller: NVIDIA Corporation GK110GL [Tesla K20c] (rev a1)

```

FIGURE 3.10: NVIDIA GRID K2 on compute

to find the most efficient to enhance the virtual desktop Performance setting. So we designed a few experiments.

- Through the host machine and the virtual machine at the same time the implementation of the ping command, and record the return value of the period of time, through the return value to determine whether the network in the virtual machine will be delayed and so on the network.
- Through the Heaven benchmark software to test the three cases under the GPU on the machine's display performance is affected, three cases are as follows 1.Install the GPU on the physical machine; 2.Through the OpenStack open and through PCI pass-through get GPU resources of the virtual machine (Hypervisor for XenServer); 3.Directly through XenServer to open and through PCI pass-through GPU resources to get the virtual machine.

In the above three cases to do the test, and in the virtual machine part, we will set the memory 8G, and virtual CPU part will be divided into

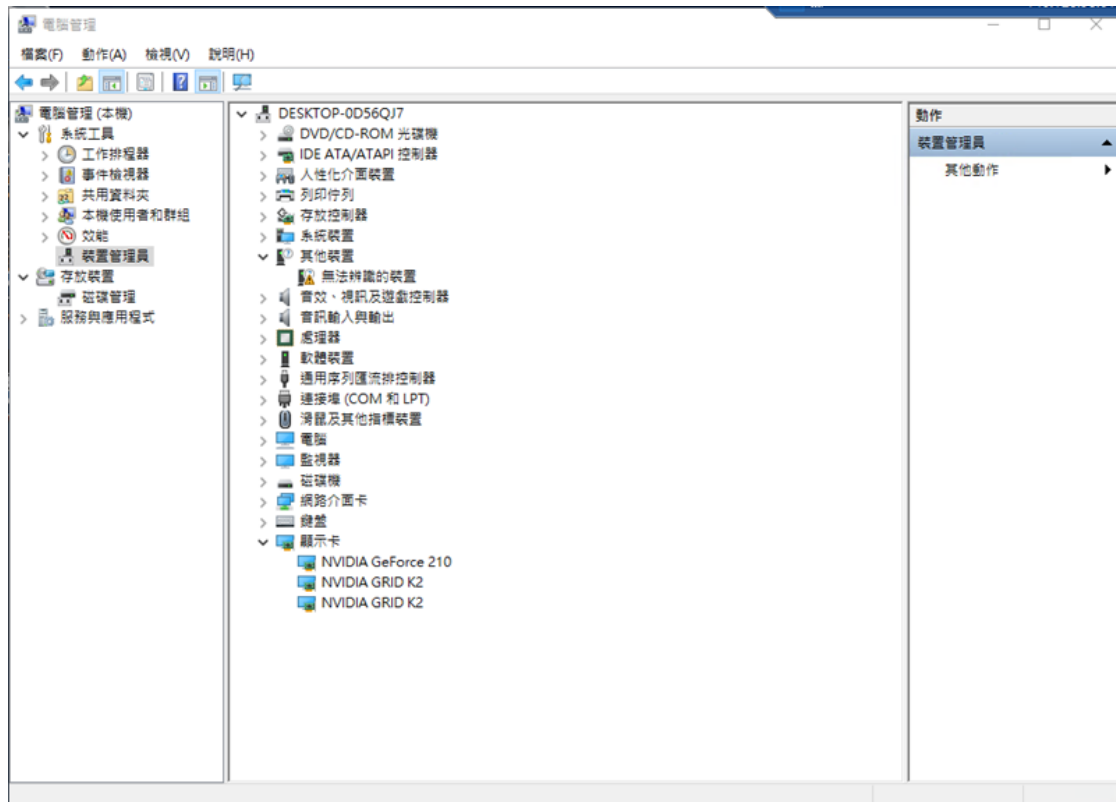


FIGURE 3.11: NVIDIA GRID K2 on Windows VM

three parts, 2socket with 2core per socket (4 vCPU), 2socket with 4core per Socket (8 vCPU), 2socket with 4core per socket (16 vCPU). And through the Heaven benchmark to get four results, respectively, Score, FPS, MinFPS, MaxFPS.

- The third experiment we will use CineBench Benchmark tool for testing, CineBench test can be obtained two return value, respectively, with the GPU-related OpenGL FPS, the other is CPU-related CPU Score, so we will CindeBenchmark to test, test the situation as the second experiment, respectively, in three different cases to do the test, and part of the virtual machine will be tested in three different CPU number, then the two return value , To determine whether the GPU can be accelerated in the virtual machine effect.

Chapter 4

Experimental Results

In this section, we will show the environment of the experiment and the results of the experiment. In subsection 4.1, we will describe our experimental environment, subsection 4.2 - 4.6 will introduce our experimental results.

4.1 Experimental Environment

We used OpenStack to build our cloud platform, which then was used to create and manage the storage distribution. As a simple example, we integrated two heterogeneous storage technologies. And we built the storage system by some VMs, in which HDFS was constructed by three VMs with specifications of 4-core CPU, 4 GB memory, and a total of 200 GB storage space. Table 4.1, 4.2, 4.3 are our experimental environment specification.

TABLE 4.1: Hardware Specification

Host name	CPU	Memory	Disk	GPU	OS
Openstack Controller	12 cores	64GB	2TB		Ubuntu 16.04.02
Openstack Network	24 cores	64GB	2TB		Ubuntu 16.04.02
Openstack Compute(KVM)	20 cores	24GB	2TB	GRID K2	Ubuntu 16.04.02
XenServer	20 cores	24GB	2TB	GRID K2	XenServer 7.0
Openstack Block Storage	64 cores	48GB	8TB		Ubuntu 16.04.02

TABLE 4.2: Virtual Machine Specification

Host name	CPU	Memory	Disk	OS
Instance 01	4 cores vCPU	8GB	100GB	Windows 10
Instance 02	8 cores vCPU	8GB	100GB	Windows 10
Instance 03	16 cores vCPU	8GB	100GB	Windows 10

TABLE 4.3: Software Specification

Software	Version
OpenStack	Newton
Python	2.7.6
MariaDB	10.1.1.14

In our experiment, we will use NVIDIA K2 GPU as a device to accelerate VDI, so the experiment will be part of the main K2.

4.2 Network Delay Experiment

The first experiment, using the PING instruction, respectively, in the HOST and virtual machine at the same time PING instructions, the command is to ping 8.8.8.8, to test whether the virtual machine and the external machine between the network is delayed. We were tested for 60 seconds, 120 seconds, the experimental results shown in Figure 4.1,4.2,

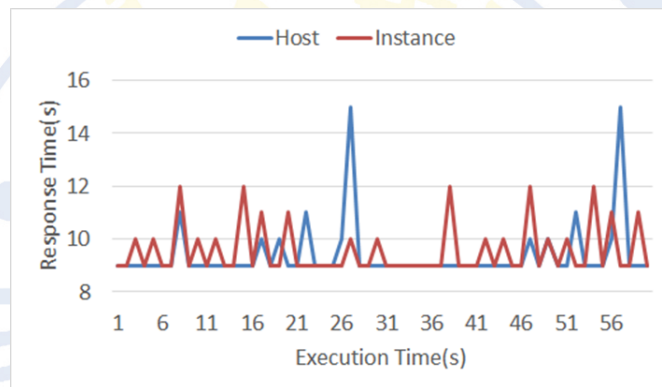


FIGURE 4.1: ping 60s



FIGURE 4.2: ping 60s

From this experiment can be found in the virtual machine and the physical machine outside the network is no significant difference, on behalf of the virtual machine will not because of OpenStack and Windows two layers of OS on the network caused by the delay, so you can ignore the experiment in the follow-up part.

4.3 Heaven Benchmark Experiment

This experiment uses Heaven Benchmark's free Benchmark tool, measures the performance of the physical machine and the virtual machine, distributes the GPU

to the virtual machine using Pass-through technology, and measures whether the vCPU will affect it by modifying the part of the virtual CPU core To the display performance of the virtual machine.

First, through the Heaven Benchmark test OpenStack open virtual machine, you can get four values, namely FPS, Min FPS, Max FPS and Score We tested three cases, namely 4vCPU, 8vCPU, 16vCPU and memory part are all 8G Ram the results as shown in Figure 4.3,4.4,4.5,4.6.

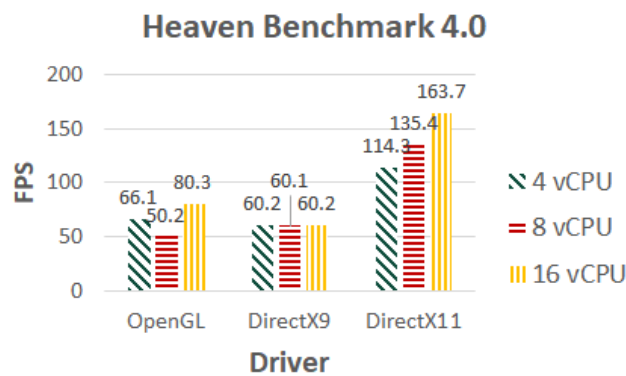


FIGURE 4.3: OpenStack VM HeavenBenchmark FPS

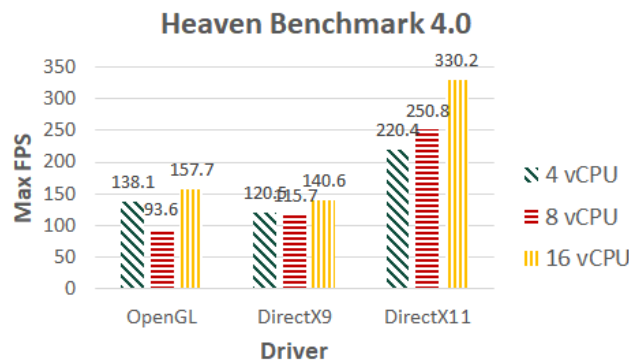


FIGURE 4.4: OpenStack VM HeavenBenchmark MaxFPS

The four charts are FPS, MAX FPS, MIN FPS, Score, after the data into a chart can be found DirectX9 FPS average are similar in the fixed memory situation, the number of virtual CPU will not Affect the performance of DirectX9, and the average FPS falls on about sixty. And DirectX11 because the CPU core of different, affecting his performance, and DirectX11 performance is significantly better than the other two drivers.

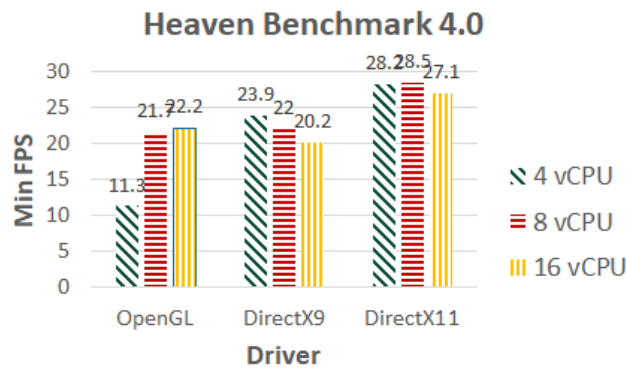


FIGURE 4.5: OpenStack VM HeavenBenchmark MinFPS

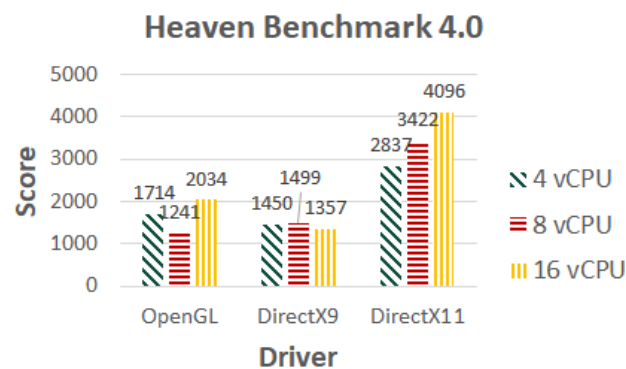


FIGURE 4.6: OpenStack VM HeavenBenchmark Score

OpenGL part is not particularly prominent performance. In addition, we found in the test, different picture resolution will also affect the performance of the screen, so we are fixed with 1280 * 720 resolution to do the test, when we use 1920 * 1080, FPS and Score will be reduced to 50% of the figures are usually between 20 to 30.

In the second part of the virtual machine test, we will compare OpenStack integration XenServer and only XenServer environment, by modifying the vCPU to test performance, the experiment we will open three virtual machines in XenServer, as in the previous part of the experiment, respectively, 4 vCPU , 8vCPU, 16vCPU three virtual machines, and are configured 8G memory, through the PCI Pass-through technology K2 configuration to the virtual machine, and the previous experiment are exactly the same. After the Heaven Benchmark test to get the

following results. The 4vcpu results shown in Figure 4.7,4.9,4.10,4.8, 8vcpu results shown in Figure 4.11,4.13,4.14,4.12, the 16vcpu results shown in Figure 4.15,4.17,4.18,4.16.

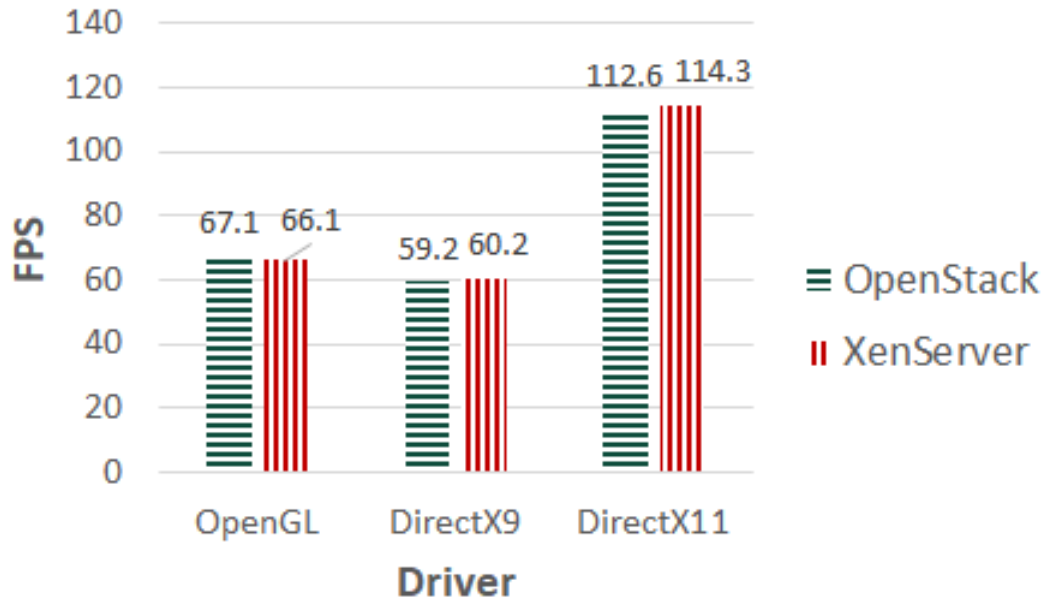


FIGURE 4.7: OpenStack and XenServer instance performance compare(4 vcpu)

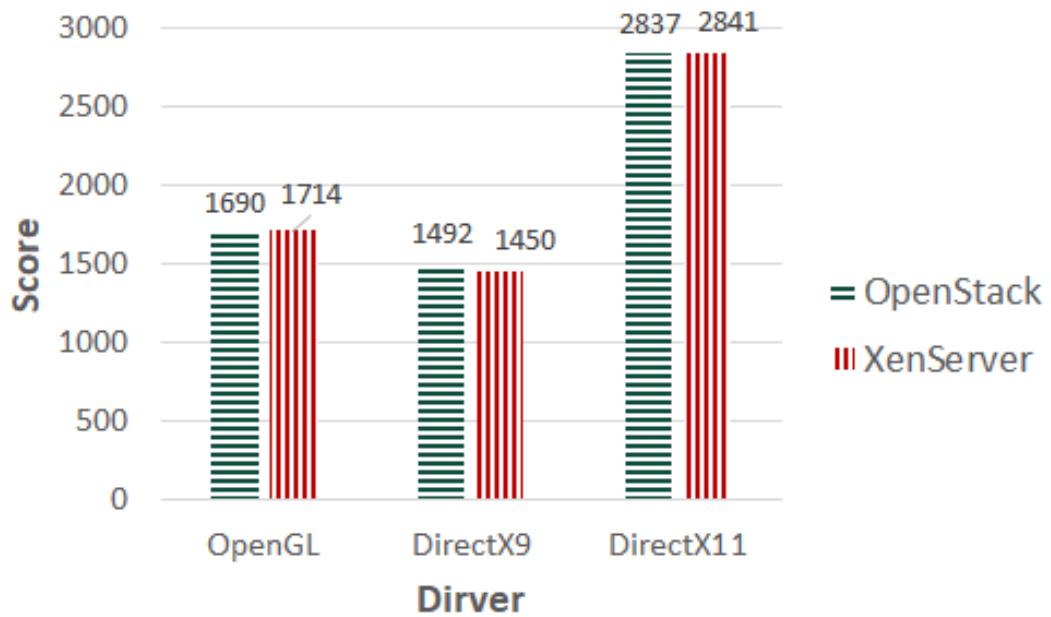


FIGURE 4.8: OpenStack and XenServer instance performance compare(4 vcpu)

Through this second experiment test results, we can see that the part of the Heaven Benchmark test is still the best performance of DirectX11, OpenGL and

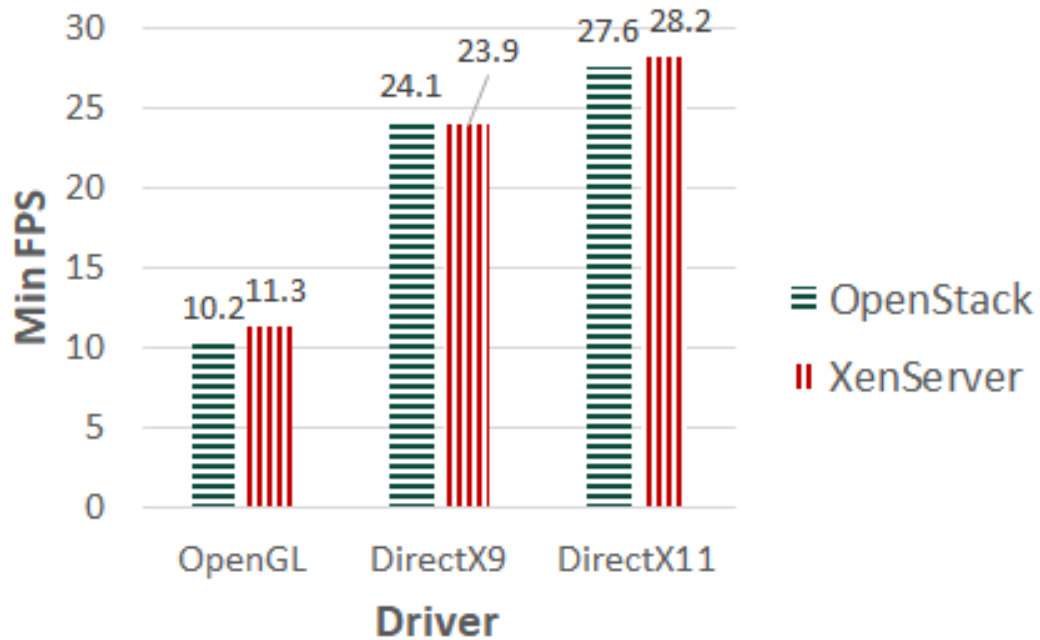


FIGURE 4.9: OpenStack and XenServer instance performance compare(4 vcpu)

DirectX11 because of the number of CPU and the impact of the test results are good or bad, and DirectX9 will not, In the case of fixed Ram, DirectX11 performance is much higher than the other two, about 2 to 3 times, the results through openStack open virtual machine test results are no different.

Through 4.19,4.20,4.21,4.22, the above comparison chart, we can find in the virtual machine performance test part, through OpenStack or OpenStack directly through the virtual machine between the two data between the difference is very small, it can be said that there is almost no gap, so We can understand that OpenStack does not affect the performance of virtual desktop display, the reason should be for our experimental environment will OpenStack and XenServer integration, which OpenStack will only open the virtual machine, through NOVA-API to communicate with XenServer, Then we do not need to operate on the virtual machine through OpenStack So in the test performance, we can directly ignore the impact of OpenStack.

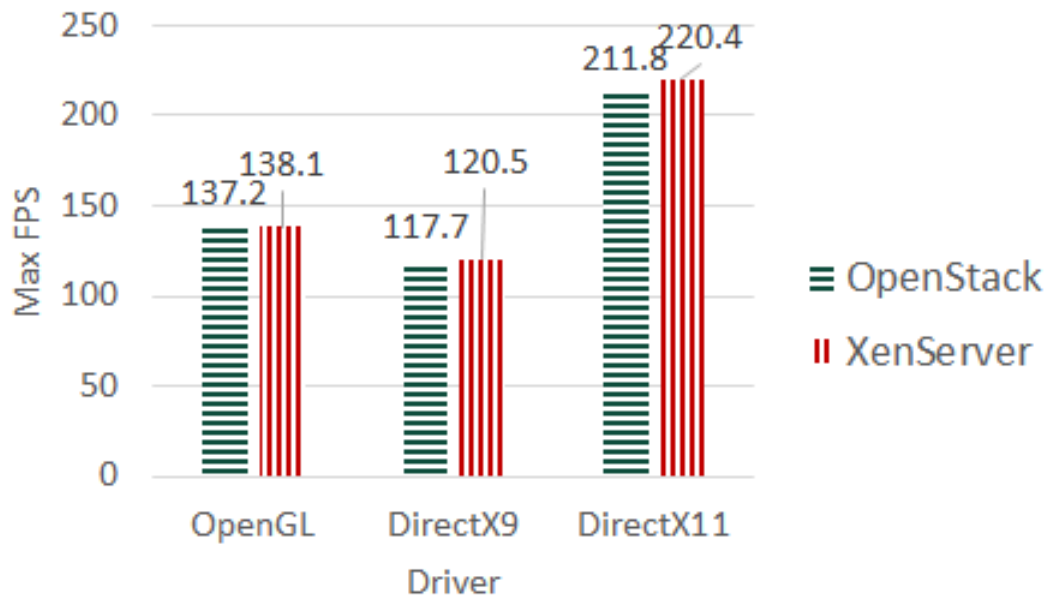


FIGURE 4.10: OpenStack and XenServer instance performance compare(4 vcpu)

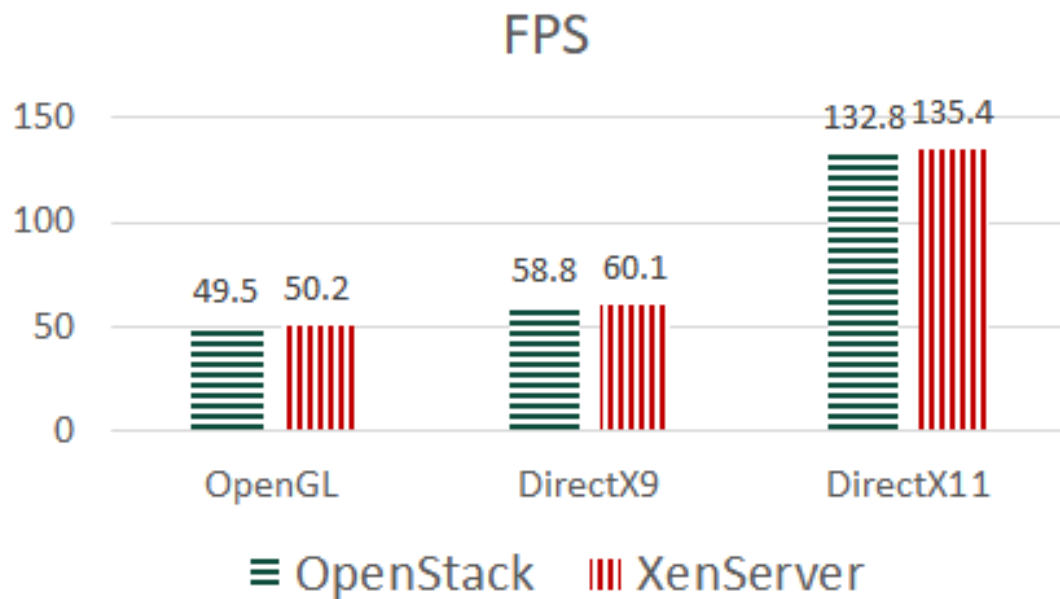


FIGURE 4.11: OpenStack and XenServer instance performance compare(8 vcpu)

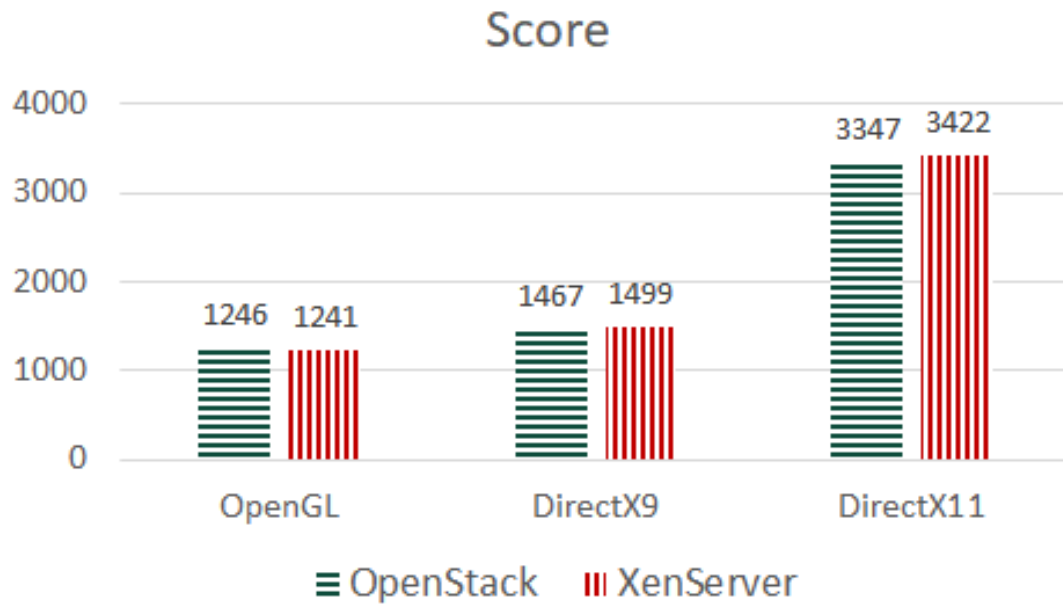


FIGURE 4.12: OpenStack and XenServer instance performance compare(8 vcpu)

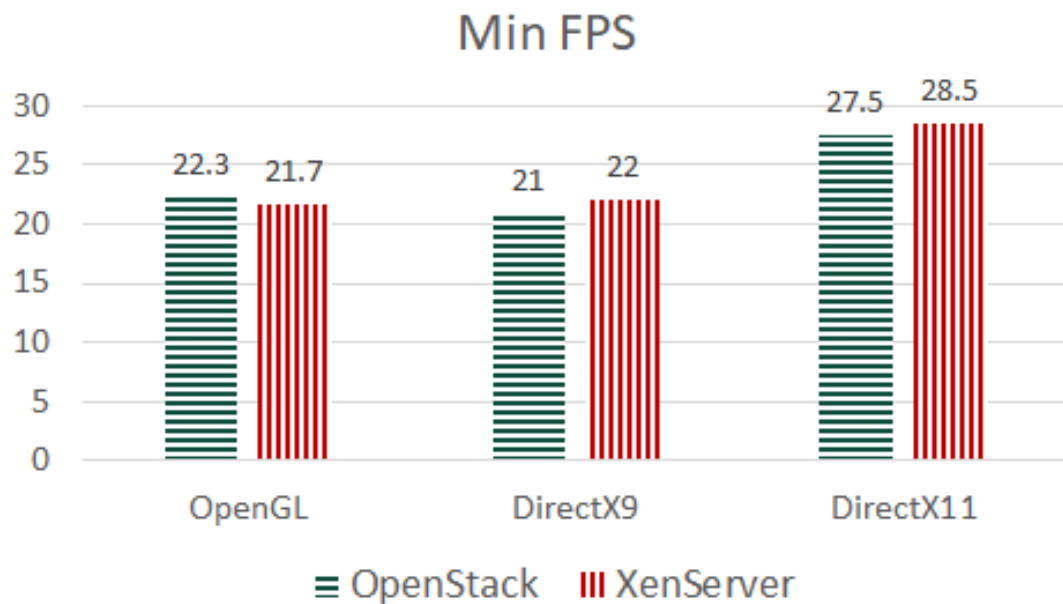


FIGURE 4.13: OpenStack and XenServer instance performance compare(8 vcpu)

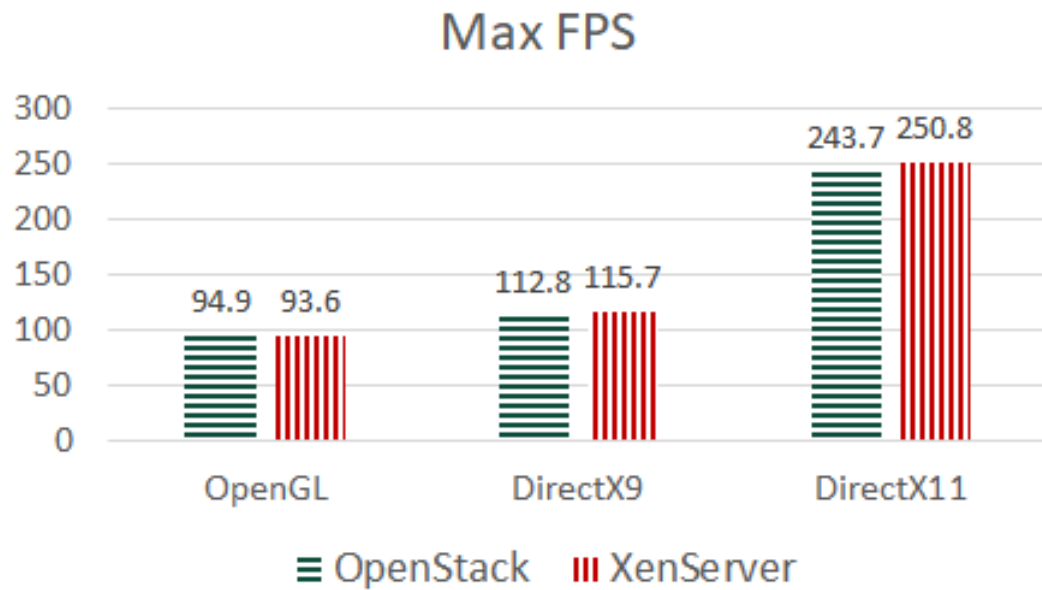


FIGURE 4.14: OpenStack and XenServer instance performance compare(8 vcpu)

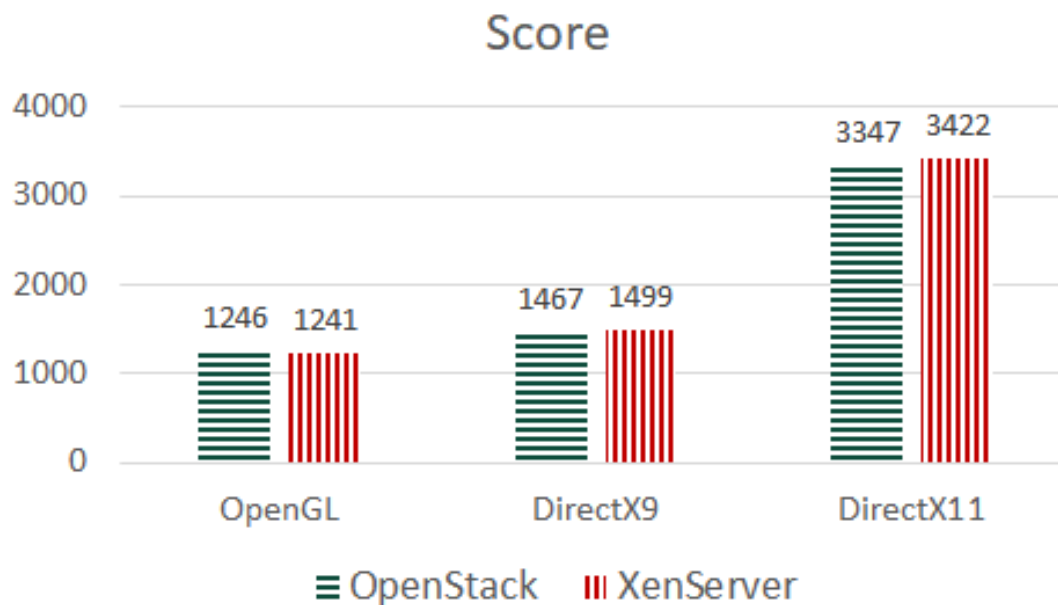


FIGURE 4.15: OpenStack and XenServer instance performance compare(16 vcpu)

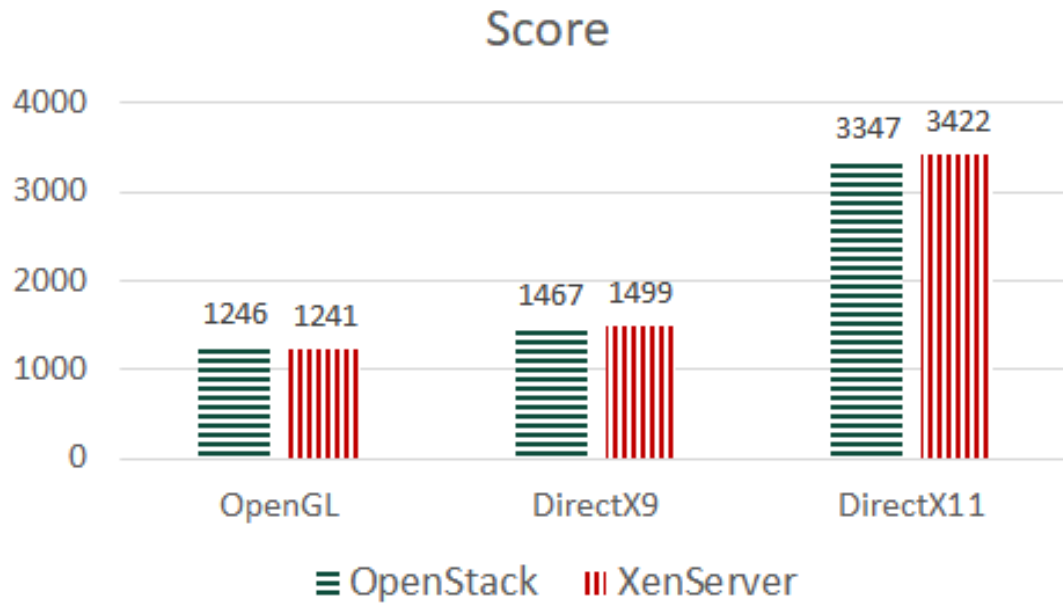


FIGURE 4.16: OpenStack and XenServer instance performance compare(16 vcpu)

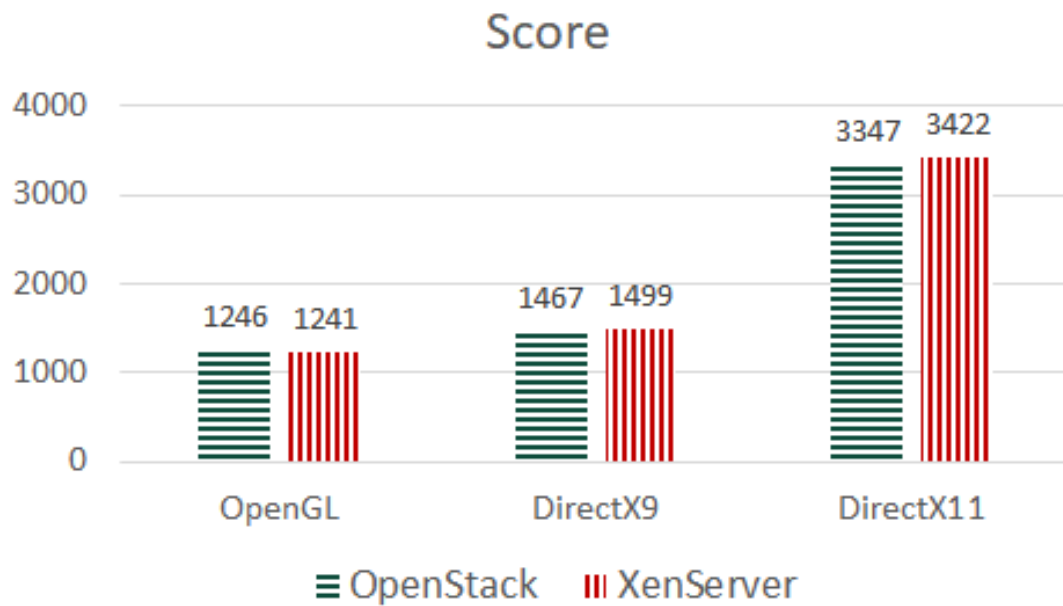


FIGURE 4.17: OpenStack and XenServer instance performance compare(16 vcpu)

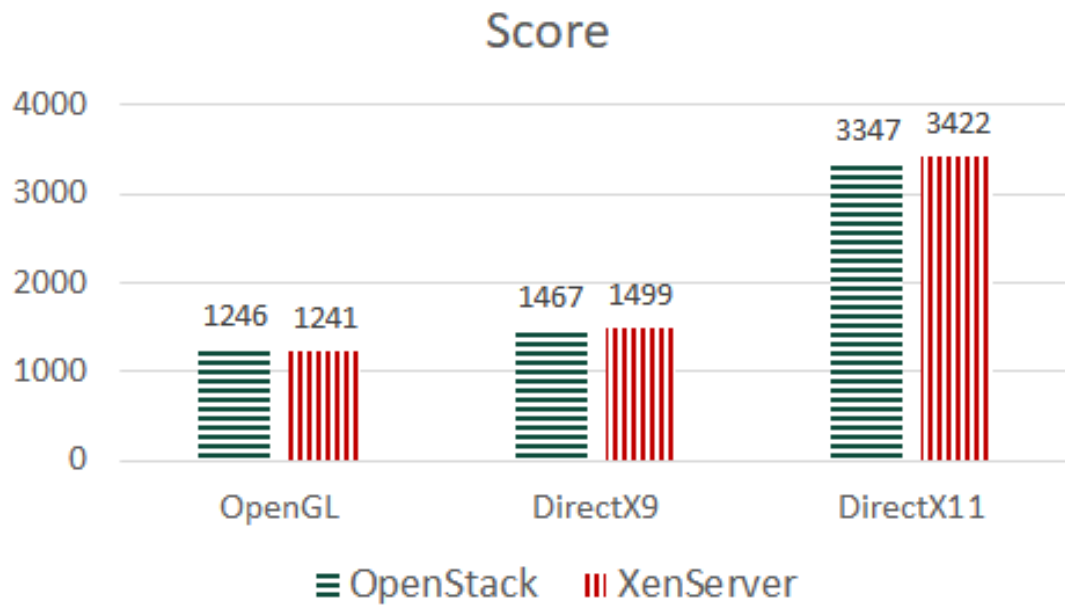


FIGURE 4.18: OpenStack and XenServer instance performance compare(16 vcpu)

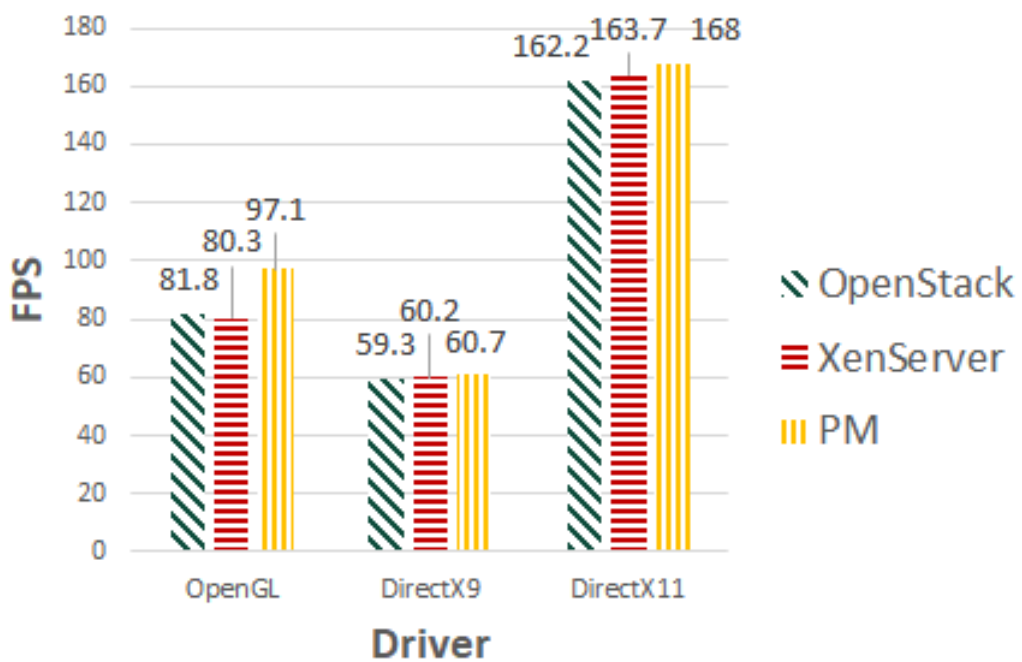


FIGURE 4.19: Compare Physical FPS

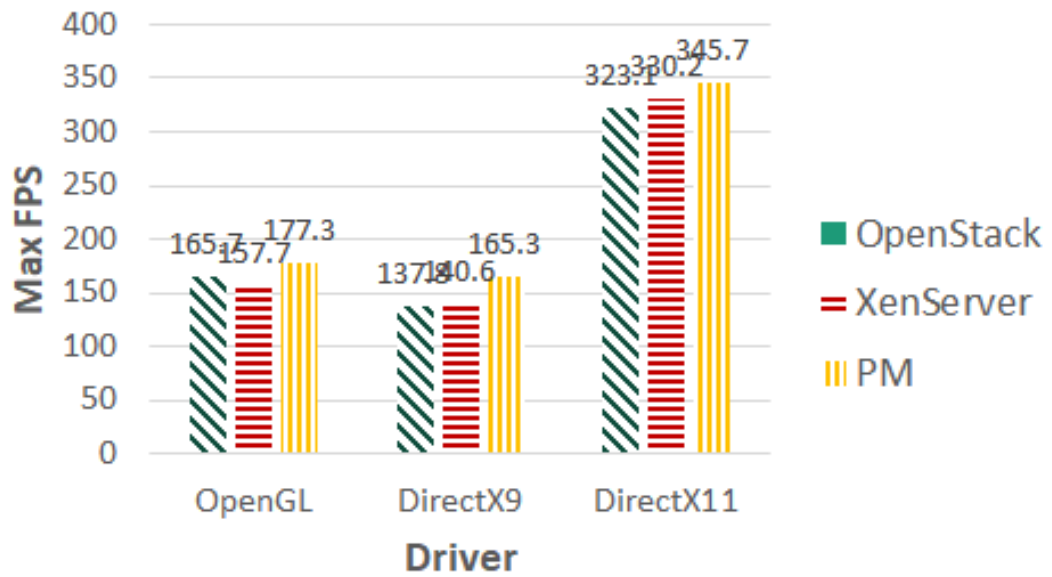


FIGURE 4.20: Compare Physical Max FPS

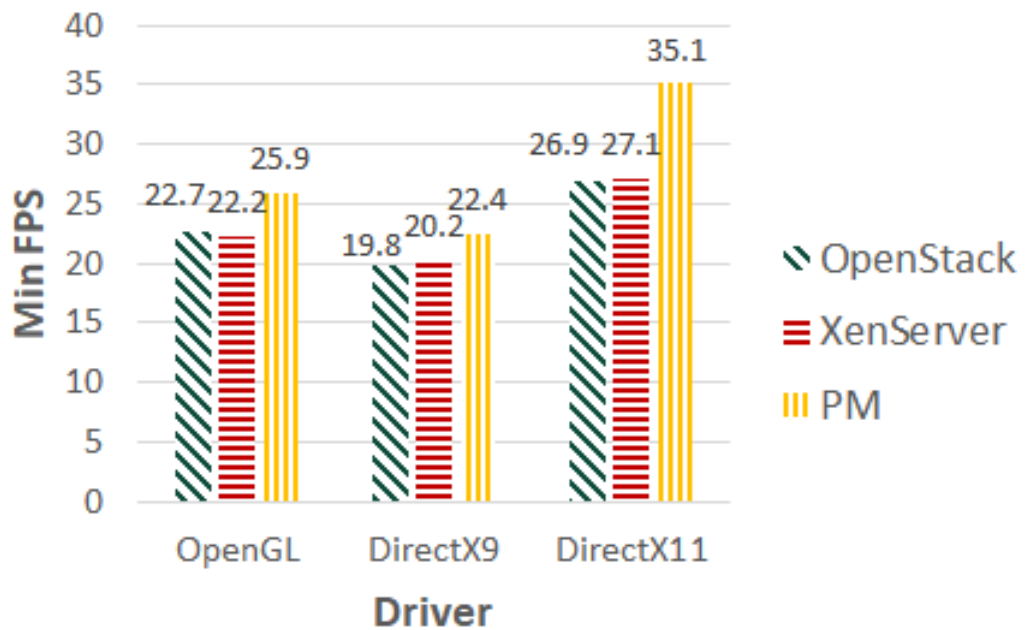


FIGURE 4.21: Compare Physical Min FPS

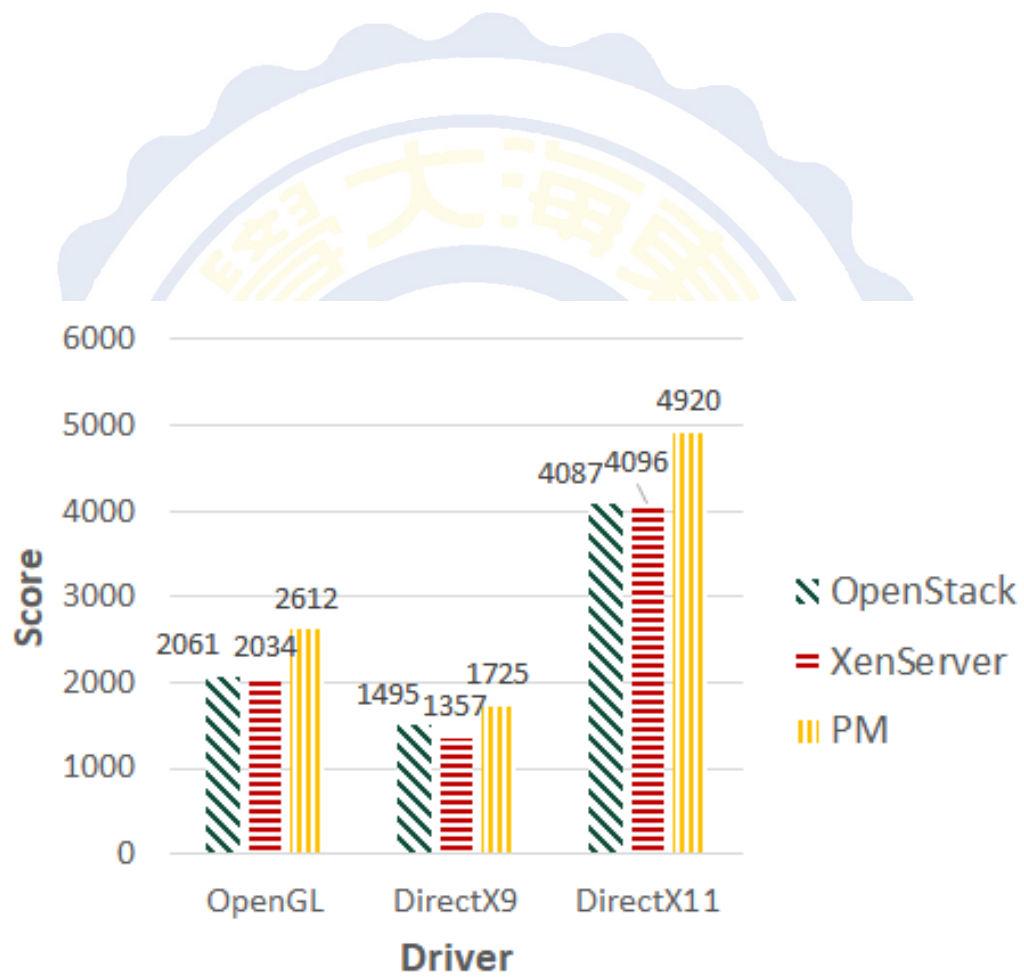


FIGURE 4.22: Compare Physical score

4.4 CineBench Experiment

In this experiment, we through the CineBench Benchmark tool to the virtual machine to do the experiment, the software will be tested through the graphical test CPU and GPU, we will be the same as the previous experiment is divided into three virtual machines, 4 vCPU, 8 VCPU, 16 vCPU, three virtual machines are fixed to 8G of memory.

The results of the test are as follows 4.23,4.24.

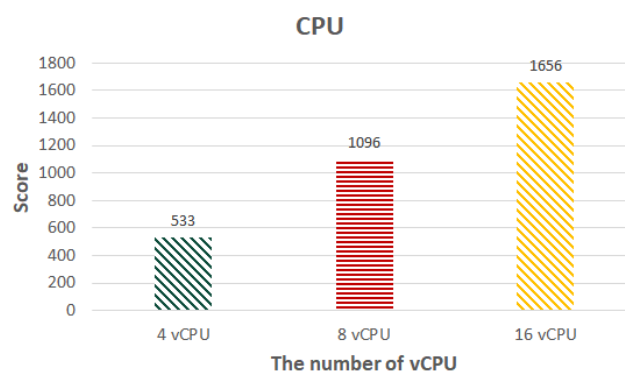


FIGURE 4.23: CineBench CPU

Through the test data we can see in the CPU part of the value based on a different number of CPU has increased significantly, 8 CPU time more than 4 times the CPU about double the score, and 16 CPU and more than 8 CPU Doubled. You can see the relationship from the picture.

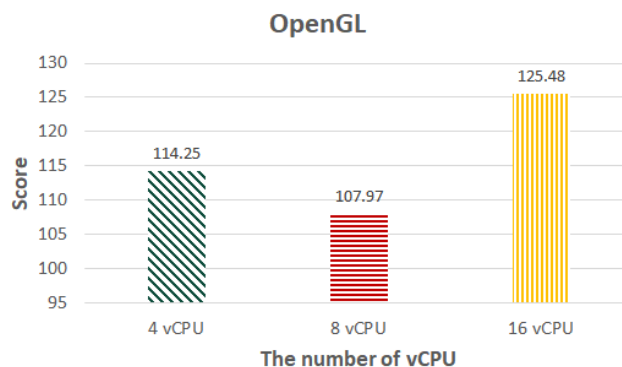


FIGURE 4.24: CineBench OpenGL

And in the OpenGL part we can find in a different number of CPU conditions, the data will not be too much difference, almost all about 120 up and down, and not too much gap, so you can know the number of CPU on the OpenGL. The effect is not so big, and this result can be verified with the results of Heaven Benchmark.

Through the above three experiments can be sorted out the conclusion of the experiment, the first point between the virtual machine and the host network is no significant difference, so the network part is almost negligible, The second point through the sky benchmark can be found in the number of different CPU CORE will indirectly affect the GPU with the effect of the screen display, you can also find different drivers will have different effects on performance, especially in the DirectX11 impact Maximum, and DirectX9 has no effect, and the worst performance. The third part of the experiment can be used by CineBench to discover the combined effects of the CPU.

Chapter 5

Conclusions and Future Work

5.1 Concluding Remark

In order to speed up the fluency of the virtual desktop and the ability to process graphics, we use Pass-Through to allow virtual machines to use the HOST's GPU resources, and through testing, including the network delay test, we learned that OpenStack virtual machine In the Neutron architecture and host machine network delay difference is not large, almost can ignore the part of the network.

In the experiment before we tested a variety of VDI, OpenStack can support GPU VDI is not much, noVNC and Spice are not normal operation, so we use the RDP as our experiment VDI. In the selected VDI after Benchmark experiment, we CPU and GPU part of the test and comparison, in different Benchmark tool, DirectX11 is the most compatible with NVIDIA GRID K2, and DirectX9 is the most inefficient Well, and the number of CPUs on DirectX9 has no effect. Which part of the performance of DirectX11 than DirectX9 and OpenGL both about 2 to 3 times the performance, including Min FPS, Max FPS, FPS AVS.

In the course of the experiment, we also found that if the entity is displayed directly through other graphics cards, K2 will not be used, but if the connection to the solid machine through RDP, K2 will begin operation, during which 10 to 25% of the upper and lower utilization rate to display the action, it can be

found through the RDP GPU will indeed accelerate the display function, but also because the GPU for VDI acceleration function, it will find through RDP test performance than directly Solid machine test on the above worse, almost worse to about 20. And through the RDP connection test, the GPU's temperature rise is also faster than the direct operation of the solid machine. Although the performance is not as good as the actual operation of the machine, but the GPU can indeed accelerate the performance of VDI, GPU can be identified for the virtual desktop can significantly improve the use of the use of experience.

5.2 Future Works

In this paper, XenServer for GPU Pass-Through experiment, the future hope to replace the different hypervisors, such as VMware ESXI or KVM, and so different Hypervisor, and through Benchmark to find out what kind of hypervisor on the GPU Pass-through support the best. The Can get the best performance.

To balance the amount of storage resources, we propose a mechanism for heterogeneous storage. In the proposed mechanism, different cluster can be integrated in our system and each file can be split into suitable storage. Besides, this mechanism has scalability so that one can add more heterogeneous storages.

OpenStack does not support vGPU mode, only through the Pass-Through, but this approach may cause a waste of GPU resources, because the GPU performance is powerful, just a virtual desktop may not play to all the features, So the future if the GPU can be through virtualization technology, assigned to a different virtual machine, so that each virtual machine can get different virtual machine resources, may be a better practice.

So in the future we plan to achieve vGPU in different ways, including the existing VMware full virtualization, XenServer vGPU, KVM KVM-GT, etc., these are the future we plan to try, the above vGPU function into OpenStack will then find out where the performance of virtual desktops can reach the maximum value,

and there will be no waste of additional performance, which is our next stage of the goal.



References

- [1] M. Zakarya and L. Gillam. Energy efficient computing, clusters, grids and clouds: A taxonomy and survey. *Sustainable Computing: Informatics and Systems*, 14:13–33, 2017.
- [2] P. Mishra, E.S. Pilli, V. Varadharajan, and U. Tupakula. Intrusion detection techniques in cloud environment: A survey. *Journal of Network and Computer Applications*, 77:18–47, 2017.
- [3] H.-J. Hong, P.-H. Tsai, and C.-H. Hsu. Dynamic module deployment in a fog computing platform. 2016.
- [4] I. Kamel, A.M. Talha, and Z.A. Aghbari. Dynamic spatial index for efficient query processing on the cloud. *Journal of Cloud Computing*, 6(1), 2017.
- [5] A. Razaque and S.S. Rizvi. Privacy preserving model: a new scheme for auditing cloud stakeholders. *Journal of Cloud Computing*, 6(1), 2017.
- [6] M. Abu Sharkh, A. Shami, and A. Ouda. Optimal and suboptimal resource allocation techniques in cloud computing data centers. *Journal of Cloud Computing*, 6(1), 2017.
- [7] A. Solano, R. Dormido, N. Duro, and J.M. Sánchez. A self-provisioning mechanism in openstack for iot devices. *Sensors (Switzerland)*, 16(8), 2016.
- [8] D. Freet, R. Agrawal, J.J. Walker, and Y. Badr. Open source cloud management platforms and hypervisor technologies: A review and comparison. volume 2016-July, 2016.

- [9] V.G. Chamorro, C.N. Castillo, and F. Lopez-Pires. An elastic voip solution based on openstack. pages 43–47, 2016.
- [10] S. Sotiriadis and N. Bessis. An inter-cloud bridge system for heterogeneous cloud platforms. *Future Generation Computer Systems*, 54:180–194, 2016.
- [11] I. Benatia, M. Ridida, H. Bendjenna, and S.B. Eom. Implementing a cloud-based decision support system in a private cloud: The infrastructure and the deployment process. *International Journal of Decision Support System Technology*, 8(1):25–42, 2016.
- [12] C. de Alfonso, A. Calatrava, and G. Moltó. Container-based virtual elastic clusters. *Journal of Systems and Software*, 127:1–11, 2017.
- [13] A. Pietrabissa, F.D. Priscoli, A. Di Giorgio, A. Giuseppi, M. Panfili, and V. Suraci. An approximate dynamic programming approach to resource management in multi-cloud scenarios. *International Journal of Control*, 90(3): 508–519, 2017.
- [14] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba. Topology-aware prediction of virtual network function resource requirements. *IEEE Transactions on Network and Service Management*, 14(1):106–120, 2017.
- [15] C.-T. Yang, J.-C. Liu, S.-T. Chen, and K.-L. Huang. Virtual machine management system based on the power saving algorithm in cloud. *Journal of Network and Computer Applications*, 80:165–180, 2017.
- [16] M. Marks and E. Niewiadomska-Szynkiewicz. Hybrid cpu/gpu platform for high performance computing. pages 508–514, 2014.
- [17] S. Boob, H. González-Vélez, and A.M. Popescu. Automated instantiation of heterogeneous fast flow cpu/gpu parallel pattern applications in clouds. pages 162–169, 2014.
- [18] B. Kim and B. Lee. Integrated management system for distributed micro-datacenters. volume 2016-March, pages 466–469, 2016.

- [19] A. Paradowski, L. Liu, and B. Yuan. Benchmarking the performance of openstack and cloudstack. pages 405–412, 2014.
- [20] J. Zhang, S. Han, J. Wan, B. Zhu, L. Zhou, Y. Ren, and W. Zhang. Im-dedup: An image management system based on deduplication applied in dwsns. *International Journal of Distributed Sensor Networks*, 2013, 2013.
- [21] M. Izumi and K. Horikawa. Toward practical use of virtual smartphone. 2012.
- [22] A. Malik, J. Ahmed, J. Qadir, and M.U. Ilyas. A measurement study of open source sdn layers in openstack under network perturbation. *Computer Communications*, 102:139–149, 2017.
- [23] R. Pacevič and A. Kačeniauskas. The development of vislt visualization service in openstack cloud infrastructure. *Advances in Engineering Software*, 103:46–56, 2017.
- [24] A. Kačeniauskas, R. Pacevič, V. Starikovičius, A. Maknickas, M. Staškūnienė, and G. Davidavičius. Development of cloud services for patient-specific simulations of blood flows through aortic valves. *Advances in Engineering Software*, 103:57–64, 2017.
- [25] R. Pacevic and A. Kaceniauskas. Deployment of visualization software and gpu rendering on an openstack cloud infrastructure. *Civil-Comp Proceedings*, 107, 2015.
- [26] Y. Yamato. Optimum application deployment technology for heterogeneous iaas cloud. *Journal of Information Processing*, 25:56–58, 2017.
- [27] Y. Yamato. Proposal of optimum application deployment technology for heterogeneous iaas cloud. pages 34–37, 2016.

Appendix A

OpenStack Installation

OpenStack Controller Installation Shell

```
#!/bin/sh
read -p "Please input your MYSQL Password: " password # 提示使用者輸入
read -p "Please input your controller ip: " controller
sudo apt-get install -y software-properties-common
sudo add-apt-repository -y cloud-archive:newton
sudo apt-get update && sudo apt-get -y dist-upgrade
sudo apt-get install -y python-openstackclient
sudo apt-get install -y mariadb-server python-pymysql

touch /etc/mysql/mariadb.conf.d/openstack.cnf
echo "[mysqld]
bind-address = ${controller}

default-storage-engine = innodb
innodb_file_per_table
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8" >> /etc/mysql/mariadb.conf.d/openstack.cnf

sudo service mysql restart
sudo mysql_secure_installation

sudo apt-get install -y rabbitmq-server
sudo rabbitmqctl add_user openstack ${password}
sudo rabbitmqctl set_permissions openstack ".*" ".*" ".*"
sudo apt-get install -y memcached python-memcache

sudo sed -i 's@-l 127.0.0.1@-l ${controller}@' /etc/memcached.conf
sudo service memcached restart
```

```
mysql -u root -p${password} -e "drop database keystone;"
mysql -u root -p${password} -e "drop database glance;"
mysql -u root -p${password} -e "drop database nova;"
mysql -u root -p${password} -e "drop database nova_api;"
mysql -u root -p${password} -e "drop database neutron;"
mysql -u root -p${password} -e "drop database cinder;"

mysql -u root -p${password} -e "CREATE DATABASE keystone;"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'
    localhost' IDENTIFIED BY '${password}';"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%
    IDENTIFIED BY '${password}';"

mysql -u root -p${password} -e "CREATE DATABASE glance;"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'
    localhost' IDENTIFIED BY '${password}';"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%
    IDENTIFIED BY '${password}';"

mysql -u root -p${password} -e "CREATE DATABASE nova;"
mysql -u root -p${password} -e "CREATE DATABASE nova_api;"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost'
    IDENTIFIED BY '${password}';"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%
    IDENTIFIED BY '${password}';"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'
    localhost' IDENTIFIED BY '${password}';"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%
    IDENTIFIED BY '${password}';"

mysql -u root -p${password} -e "CREATE DATABASE neutron;"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'
    localhost' IDENTIFIED BY '${password}';"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%
    IDENTIFIED BY '${password}';"

mysql -u root -p${password} -e "CREATE DATABASE cinder;"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'
    localhost' IDENTIFIED BY '${password}';"
mysql -u root -p${password} -e "GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%
    IDENTIFIED BY '${password}';"
```

```
echo "manual" | sudo tee /etc/init/keystone.override
sudo apt-get install keystone apache2 libapache2-mod-wsgi -y

sudo sed -i 's@#admin_token = <None>@admin_token =21
    d7fb48086e09f30d40be5a5e95a7196f2052b2cae6b491@' /etc/keystone/keystone.conf
sudo sed -i 's~connection = sqlite:////var/lib/keystone/keystone.db~connection = mysql
    +pymysql://keystone:'${password}'@'${controller}'/keystone~' /etc/keystone/
    keystone.conf
sudo sed -i 's@#servers = localhost:11211@servers = '${controller}':11211@' /etc/
    keystone/keystone.conf
sudo sed -i 's@#provider = uuid@provider = fernet@' /etc/keystone/keystone.conf

sudo keystone-manage db_sync
sudo keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
sudo keystone-manage credential_setup --keystone-user keystone --keystone-group
    keystone

sudo keystone-manage bootstrap --bootstrap-password ${password} \
--bootstrap-admin-url http://${controller}:35357/v3/ \
--bootstrap-internal-url http://${controller}:35357/v3/ \
--bootstrap-public-url http://${controller}:5000/v3/ \
--bootstrap-region-id RegionOne

echo "ServerName ${controller}">> /etc/apache2/apache2.conf
sudo ln -s /etc/apache2/sites-available/keystone.conf /etc/apache2/sites-enabled
sudo service apache2 restart
sudo rm -f /var/lib/keystone/keystone.db

export OS_USERNAME=admin
export OS_PASSWORD=${password}
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://${controller}:35357/v3
export OS_IDENTITY_API_VERSION=3

openstack project create --domain default \
--description "Service Project" service
openstack project create --domain default \
--description "Demo Project" demo
openstack user create --domain default --password ${password} demo

openstack role create user
openstack role add --project demo --user demo user
```

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=${password}
export OS_AUTH_URL=http://${controller}:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2

openstack user create --domain default --password ${password} --email glance@example.com glance

openstack role add --project service --user glance admin

openstack service create --name glance --description "OpenStack Image service" image

openstack endpoint create --region RegionOne \
image public http://${controller}:9292

openstack endpoint create --region RegionOne \
image internal http://${controller}:9292

openstack endpoint create --region RegionOne \
image admin http://${controller}:9292

sudo apt-get install -y glance
sudo sed -i 's~sqlite_db = /var/lib/glance/glance.sqlite~connection = mysql+pymysql://glance:${password}@${controller}/glance~/ /etc/glance/glance-api.conf
sudo sed -i 'N; s@[keystone_authtoken]\n@[keystone_authtoken]\nauth_uri = http://${controller}:5000\nauth_url = http://${controller}:35357\nmemcached_servers = ${controller}:11211\nauth_type = password\nproject_domain_name = default\nuser_domain_name = default\nproject_name = service\nusername = glance\npassword = ${password}@' /etc/glance/glance-api.conf
sudo sed -i 's@#flavor = keystone@flavor = keystone@' /etc/glance/glance-api.conf
sudo sed -i 's@#stores = file ,http@stores = file ,http@' /etc/glance/glance-api.conf
sudo sed -i 's@#default_store = file@default_store = file@' /etc/glance/glance-api.conf
sudo sed -i 's@#filesystem_store_datadir = /var/lib/glance/images@filesystem_store_datadir = /var/lib/glance/images/@' /etc/glance/glance-api.conf

sudo sed -i 's~sqlite_db = /var/lib/glance/glance.sqlite~connection = mysql+pymysql://glance:${password}@${controller}/glance~/ /etc/glance/glance-registry.conf
```

```
sudo sed -i 'N; s@[keystone_authtoken\]@\[keystone_authtoken\]\nauth_uri = http://'\${
  controller}':5000\nauth_url = http://'\${controller}':35357\nmemcached_servers = '\$
  {controller}':11211\nauth_type = password\nproject_domain_name = default\
  nuser_domain_name = default\nproject_name = service\nusername = glance\npassword =
  '\${password}'@' /etc/glance/glance-registry.conf
sudo sed -i 's@#flavor = keystone@flavor = keystone@' /etc/glance/glance-registry.conf

sudo glance-manage db_sync
sudo service glance-registry restart
sudo service glance-api restart
sudo rm -f /var/lib/glance/glance.sqlite

sudo wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img

openstack image create "cirros-0.3.4-x86_64" \
--file cirros-0.3.4-x86_64-disk.img \
--disk-format qcow2 --container-format bare \
--public

export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=${password}
export OS_AUTH_URL=http://\${controller}:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2

openstack user create --domain default --password \${password} --email nova@example.com
nova
openstack role add --project service --user nova admin
openstack service create --name nova --description "OpenStack Compute" compute

openstack endpoint create --region RegionOne \
compute public http://\${controller}:8774/v2.1/%\((tenant_id\)s

openstack endpoint create --region RegionOne \
compute internal http://\${controller}:8774/v2.1/%\((tenant_id\)s

openstack endpoint create --region RegionOne \
compute admin http://\${controller}:8774/v2.1/%\((tenant_id\)s

sudo apt-get install -y nova-api nova-conductor nova-consoleauth \
nova-novncproxy nova-scheduler
```



```
sudo sed -i 'N; s@[DEFAULT]\n@[DEFAULT]\n\nmy_ip = '${controller}'\nrpc_backend =
    rabbit\nauth_strategy = keystone\nuse_neutron = True\nfirewall_driver = nova.virt.
    firewall.NoopFirewallDriver\n@' /etc/nova/nova.conf

sudo sed -i 'N; s~\[database\]\n~\[database\]\nconnection = mysql+pymysql://nova:'${
    password}'@${controller}/nova\n#~' /etc/nova/nova.conf
sudo sed -i 'N; s~\[api_database\]\n~\[api_database\]\nconnection = mysql+pymysql://
    nova:'${password}'@${controller}/nova_api\n#~' /etc/nova/nova.conf
sudo sed -i 's@lock_path=/var/lock/nova@lock_path=/var/lib/nova/tmp@' /etc/nova/nova.
    conf

echo "
[vnc]
vncserver_listen = ${controller}
vncserver_proxyclient_address = ${controller}

[oslo_messaging_rabbit]
rabbit_host = ${controller}
rabbit_userid = openstack
rabbit_password = ${password}

[keystone_authtoken]
auth_uri = http://${controller}:5000
auth_url = http://${controller}:35357
memcached_servers = ${controller}:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = ${password}

[glance]
api_servers = http://${controller}:9292

" >> /etc/nova/nova.conf

sudo nova-manage api_db sync
sudo nova-manage db sync
```

```
sudo service nova-api restart
sudo service nova-consoleauth restart
sudo service nova-scheduler restart
sudo service nova-conductor restart
sudo service nova-novncproxy restart

sudo rm -f /var/lib/nova/nova.sqlite
openstack compute service list

export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=${password}
export OS_AUTH_URL=http://${controller}:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2

openstack user create --domain default --password ${password} --email neutron@example.
com neutron

openstack role add --project service --user neutron admin

openstack service create --name neutron --description "OpenStack Networking" network

openstack endpoint create --region RegionOne \
network public http://${controller}:9696

openstack endpoint create --region RegionOne \
network internal http://${controller}:9696

openstack endpoint create --region RegionOne \
network admin http://${controller}:9696

sudo apt-get install -y neutron-server neutron-plugin-ml2

sudo sed -i 's@#service_plugins =@service_plugins =router@' /etc/neutron/neutron.conf
sudo sed -i 's@#allow_overlapping_ips = false@allow_overlapping_ips = True@' /etc/
neutron/neutron.conf
sudo sed -i 's@#rpc_backend = rabbit@rpc_backend = rabbit@' /etc/neutron/neutron.conf
sudo sed -i 's@#auth_strategy = keystone@auth_strategy = keystone@' /etc/neutron/
neutron.conf
sudo sed -i 's@#notify_nova_on_port_status_changes =
true@notify_nova_on_port_status_changes = true@' /etc/neutron/neutron.conf
sudo sed -i 's@#notify_nova_on_port_data_changes =
true@notify_nova_on_port_data_changes = true@' /etc/neutron/neutron.conf
```

```

sudo sed -i 's~connection = sqlite:////var/lib/neutron/neutron.sqlite~connection =
mysql+pymysql://neutron:'${password}'@'${controller}'/neutron~' /etc/neutron/
neutron.conf
sudo sed -i 's@#rabbit_host = localhost@rabbit_host = '${controller}'@' /etc/neutron/
neutron.conf
sudo sed -i 's@#rabbit_userid = guest@rabbit_userid = openstack@' /etc/neutron/neutron
.conf
sudo sed -i 's@#rabbit_password = guest@rabbit_password = '${password}'@' /etc/neutron
/neutron.conf
sudo sed -i 's@# From keystonemiddleware.auth_token@# From keystonemiddleware.
auth_token\nauth_uri = http://'${controller}':5000\nauth_url = http://'${
controller}':35357\nmemcached_servers = '${controller}':11211\nauth_type =
password\nproject_domain_name = default\nuser_domain_name = default\nproject_name
= service\nusername = neutron\npassword = '${password}'\n@' /etc/neutron/neutron.
conf
sudo sed -i 'N; s@[nova\\]\n@[nova\\]\n\nauth_url = http://'${controller}':35357\
nauth_type = password\nproject_domain_name = default\nuser_domain_name = default\
nregion_name = RegionOne\nproject_name = service\nusername = nova\npassword = '${
password}'@' /etc/neutron/neutron.conf
sudo sed -i 's@#type_drivers = local , flat , vlan , gre , vxlan , geneve@type_drivers = flat ,
vlan , gre , vxlan@' /etc/neutron/plugins/ml2/ml2_conf.ini
sudo sed -i 's@#tenant_network_types = local@tenant_network_types = vxlan@' /etc/
neutron/plugins/ml2/ml2_conf.ini
sudo sed -i 's@#mechanism_drivers =@mechanism_drivers =openvswitch,l2population@' /etc
/neutron/plugins/ml2/ml2_conf.ini
sudo sed -i 's@#extension_drivers =@extension_drivers = port_security@' /etc/neutron/
plugins/ml2/ml2_conf.ini

sudo sed -i 's@# VXLAN VNI IDs that are available for tenant network allocation (list
value)@# VXLAN VNI IDs that are available for tenant network allocation (list
value)\nvn_ranges = 1:1000@' /etc/neutron/plugins/ml2/ml2_conf.ini
sudo sed -i 's@#enable_ipset = true@enable_ipset = true@' /etc/neutron/plugins/ml2/
ml2_conf.ini

echo "
[neutron]
url = http://'${controller}':9696
auth_url = http://'${controller}':35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = ${password}
service_metadata_proxy = True

```

```

metadata_proxy_shared_secret = ${password}
" >> /etc/nova/nova.conf

sudo neutron-db-manage --config-file /etc/neutron/neutron.conf \
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini \
upgrade head
sudo service nova-api restart
sudo service neutron-server restart
neutron ext-list

sudo apt-get install openstack-dashboard -y

sudo sed -i 's@OPENSTACK_HOST = "127.0.0.1"@OPENSTACK_HOST = "'${controller}'"@' /etc/
openstack-dashboard/local_settings.py
sudo sed -i 's@127.0.0.1:11211@'${controller}':11211@' /etc/openstack-dashboard/
local_settings.py

echo "SESSION_ENGINE = 'django.contrib.sessions.backends.cache'" >> /etc/openstack-
dashboard/local_settings.py
sudo sed -i 's@OPENSTACK_KEYSTONE_URL = "http://%s:5000/v2.0" %
OPENSTACK_HOST@OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST@' /
etc/openstack-dashboard/local_settings.py
sudo sed -i 's@_member_@user@' /etc/openstack-dashboard/local_settings.py
sudo sed -i 's@#OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT =
False@OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True@' /etc/openstack-dashboard/
local_settings.py
sudo sed -i 's@#OPENSTACK_KEYSTONE_DEFAULT_DOMAIN @OPENSTACK_KEYSTONE_DEFAULT_DOMAIN @
' /etc/openstack-dashboard/local_settings.py
sudo sed -i 's@#OPENSTACK_API_VERSIONS@OPENSTACK_API_VERSIONS@' /etc/openstack-
dashboard/local_settings.py
sudo sed -i 's@# "identity": 3,@ "identity": 3,@' /etc/openstack-dashboard/
local_settings.py
sudo sed -i 's@# "image": 2,@ "image": 2,@' /etc/openstack-dashboard/
local_settings.py
sudo sed -i 's@# "volume": 2,@ "volume": 2,\n}@' /etc/openstack-dashboard/
local_settings.py
sudo service apache2 reload
sudo service apache2 restart

openstack flavor create --ram 1024 --disk 10 --vcpus 1 test

echo "
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin

```

```
export OS_PASSWORD=${password}
export OS_AUTH_URL=http://${controller}:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
" >> admin-openrc
```

OpenStack Network Installation Shell

```
#!/bin/sh
read -p "Please input your controller ip: " controller
read -p "Please input your network ip: " network
read -p "Please input your passwd: " password

sudo apt-get install -y software-properties-common
sudo add-apt-repository -y cloud-archive:neutron
sudo apt-get update && sudo apt-get -y dist-upgrade
echo "
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
">> /etc/sysctl.conf

sudo sysctl -p

sudo apt-get install -y neutron-plugin-ml2 neutron-l3-agent \
neutron-dhcp-agent neutron-metadata-agent \
neutron-openvswitch-agent

sudo sed -i 's@#verbose = true@verbose = true@' /etc/neutron/neutron.conf
sudo sed -i 's@#rpc_backend = rabbit@rpc_backend = rabbit@' /etc/neutron/neutron.conf
sudo sed -i 's@#auth_strategy = keystone@auth_strategy = keystone@' /etc/neutron/
neutron.conf
sudo sed -i 's@#service_plugins =@service_plugins = router@' /etc/neutron/neutron.conf
sudo sed -i 's@#allow_overlapping_ips = false@allow_overlapping_ips = True@' /etc/
neutron/neutron.conf

sudo sed -i 's@connection = sqlite:///var/lib/neutron/neutron.sqlite@#connection =
sqlite:///var/lib/neutron/neutron.sqlite@' /etc/neutron/neutron.conf
sudo sed -i 's@#rabbit_host = localhost@rabbit_host = '${controller}'@' /etc/neutron/
neutron.conf
sudo sed -i 's@#rabbit_userid = guest@rabbit_userid = openstack@' /etc/neutron/neutron
.conf
sudo sed -i 's@#rabbit_password = guest@rabbit_password = '${password}'@' /etc/neutron
/neutron.conf
```



```

sudo sed -i 'N; s@[keystone_authtoken]\n@[keystone_authtoken]\nauth_uri = http://
    ${controller}:5000\nauth_url = http://'${controller}':35357\nmemcached_servers =
    '${controller}':11211\nauth_type = password\nproject_domain_name = default\n
    nuser_domain_name = default\nproject_name = service\nusername = neutron\npassword
    = '${password}'@' /etc/neutron/neutron.conf

sudo sed -i 's@#type_drivers = local , flat , vlan , gre , vxlan , geneve@type_drivers = flat ,
    vlan , gre , vxlan@' /etc/neutron/plugins/ml2/ml2_conf.ini
sudo sed -i 's@#tenant_network_types = local@tenant_network_types = vxlan@' /etc/
    neutron/plugins/ml2/ml2_conf.ini
sudo sed -i 's@#mechanism_drivers =@mechanism_drivers =openvswitch,l2population@' /etc
    /neutron/plugins/ml2/ml2_conf.ini
sudo sed -i 's@#extension_drivers =@extension_drivers =port_security@' /etc/neutron/
    plugins/ml2/ml2_conf.ini
sudo sed -i 's@#flat_networks =@flat_networks = external\n#@' /etc/neutron/plugins/ml2
    /ml2_conf.ini
sudo sed -i 'N; s@[ml2_type_vxlan\]@\[ml2_type_vxlan\]\nvn_ranges = 1:1000@' /etc/
    neutron/plugins/ml2/ml2_conf.ini
sudo sed -i 's@#enable_ipset = true@enable_ipset = true@' /etc/neutron/plugins/ml2/
    ml2_conf.ini

sudo sed -i 's@#local_ip = <None>@local_ip = '${network}'@' /etc/neutron/plugins/ml2/
    openvswitch_agent.ini
sudo sed -i 's@#bridge_mappings =@bridge_mappings = external:br-ex@' /etc/neutron/
    plugins/ml2/openvswitch_agent.ini
sudo sed -i 's@#tunnel_types =@tunnel_types = vxlan@' /etc/neutron/plugins/ml2/
    openvswitch_agent.ini
sudo sed -i 's@#l2_population = false@l2_population = True@' /etc/neutron/plugins/ml2/
    openvswitch_agent.ini
sudo sed -i 's@#prevent_arp_spoofing = true@prevent_arp_spoofing = true@' /etc/neutron
    /plugins/ml2/openvswitch_agent.ini
sudo sed -i 's@#enable_security_group = true@enable_security_group = True@' /etc/
    neutron/plugins/ml2/openvswitch_agent.ini
sudo sed -i 's@#firewall_driver = <None>@firewall_driver = neutron.agent.linux.
    iptables_firewall.OVSHybridIptablesFirewallDriver@' /etc/neutron/plugins/ml2/
    openvswitch_agent.ini

sudo sed -i 's@#verbose = true@verbose = true@' /etc/neutron/l3_agent.ini
sudo sed -i 's@#interface_driver = <None>@interface_driver = neutron.agent.linux.
    interface.OVSInterfaceDriver@' /etc/neutron/l3_agent.ini
sudo sed -i 's@#external_network_bridge = br-ex@external_network_bridge =@' /etc/
    neutron/l3_agent.ini

sudo sed -i 's@#verbose = true@verbose = true@' /etc/neutron/dhcp_agent.ini
sudo sed -i 's@#interface_driver = <None>@interface_driver = neutron.agent.linux.
    interface.OVSInterfaceDriver@' /etc/neutron/dhcp_agent.ini
sudo sed -i 's@#dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq@dhcp_driver = neutron.
    agent.linux.dhcp.Dnsmasq@' /etc/neutron/dhcp_agent.ini

```

```
sudo sed -i 's@#enable_isolated_metadata = false@enable_isolated_metadata = True@' /
etc/neutron/dhcp_agent.ini
sudo sed -i 's@#dnsmasq_config_file =@dnsmasq_config_file = /etc/neutron/dnsmasq-
neutron.conf@' /etc/neutron/dhcp_agent.ini
echo 'dhcp-option-force=26,1450' | sudo tee /etc/neutron/dnsmasq-neutron.conf

sudo sed -i 's@#nova_metadata_ip = 127.0.0.1@nova_metadata_ip = '${controller}'@' /etc
/neutron/metadata_agent.ini
sudo sed -i 's@#metadata_proxy_shared_secret =@metadata_proxy_shared_secret = '${
password}'@' /etc/neutron/metadata_agent.ini
```

