World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:7, No:9, 2013

# Enabling Remote Desktop in a Virtualized Environment for Cloud Services

Shuen-Tai Wang, Yu-Ching Lin, and Hsi-Ya Chang

Open Science Index, Computer and Information Engineering Vol:7, No:9, 2013 publications.waset.org/16614/pdf

***Abstract*—**Cloud computing is the innovative and leading information technology model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort. This paper presents our development on enabling an individual user's desktop in a virtualized environment, which is stored on a remote virtual machine rather than locally. We present the initial work on the integration of virtual desktop and application sharing with virtualization technology. Given the development of remote desktop virtualization, this proposed effort has the potential to positively provide an efficient, resilience and elastic environment for online cloud service. Users no longer need to burden the cost of software licenses and platform maintenances. Moreover, this development also helps boost user productivity by promoting a flexible model that lets users access their desktop environments from virtually anywhere.

***Keywords*—**Cloud Computing, Virtualization, Virtual Desktop, Elastic Environment.

## I. Introduction

CLOUD computing [1], [2] is a kind of metered service using massive and scalable computing technologies via the Internet. It has to be a highly elastic environment which provides stable services to users. The users can request access from a set of web interfaces that manage and monitor a pool of computing resources including machines, network, storage, operating system, application programs and development environments. When granted service requests, a part of the computing resources in the pool is dedicated to the requesting user until those resources are released. Moreover, users can not actually see or specify the physical location and organization of the equipment hosting the resources.

To achieve these goals, adopting virtualization technology [3], [4] in cloud environment becomes a major trend. Virtualization technology acts as a central component that can achieve the purpose of cloud platforms and services, and it is a promising approach to consolidating multiple services onto a smaller number of computing resources. A virtualized server environment allows computing resources to be shared among multiple performance-isolated platforms called virtual machines [5], [6]. A virtual machine is a software implementation of a machine that executes related programs like a physical machine. Each virtual machine includes its own system kernel, OS, supporting libraries and applications. A hypervisor provides a uniform abstraction of the underlying physical machine, and multiple virtual machines can execute simultaneously on a single hypervisor. Decoupling of virtual machine from the underlying physical hardware is able to allow the same virtual machine to be started and run on different physical environments. Thus virtualization is seen as an enabler for cloud computing, allowing the cloud service provider the necessary flexibility to move and allocate the computing resources requested by the user wherever the physical resources are available.

Virtualization also enables on-demand resource provisioning model in which computing resources such as CPU, memory, and disk space are made available to applications only as needed and not allocated statically. So by dynamically provisioning virtual machines, consolidating the workload, and turning computers on and off as needed, the administrators can maintain the desired quality of service while achieving higher computer utilization.

Considerations for an individual user's desktop in such virtualized environment, the remote desktop virtualization have received great interests in virtualization research community. Many [7], [8], [9], [10] have realized the concept of desktop virtualization. We also tried many implementations before developing our virtual desktop service. VMware View [11] is a commercial desktop-virtualization product developed by VMware that allows end users to access their virtual desktop from multiple locations. VMware View can operate a range of physical devices, and the functionality of the user's virtual desktop is not dependent on the capabilities of the devices themselves. Citrix XenDesktop [12] is a suite of desktop virtualization products from Citrix Systems. Through its bundled components, XenDesktop can deliver several different types of virtual desktops. But both VMware View and XenDesktop might not be an ideal solution to be deployed by an educational institution or commercial use because the licensing costs. On the other hand, for the hypervisor, we had adopted the Kernel-based Virtual Machine (KVM) [13] as the hypervisor in our virtualized platform. But XenDesktop just can run on HyperV, XenServer, and VMWare View only runs on ESX.

In this paper, we aim at the adoption of desktop virtualization and cloud technologies. We developed a virtual desktop and application sharing system which is efficient, resilience and independent of the operating system. This system provides the application execution takes place on a remote operating system which is linked to the local client device over a network using a remote display protocol through which the user interacts with

S.T. Wang is with the National Center for High-Performance Computing, Taiwan, R.O.C. (e-mail: stwang@nchc.org.tw).
Y.C. Lin is with the National Center for High-Performance Computing, Taiwan, R.O.C. (e-mail: 1203043@nchc.org.tw).
H.Y. Chang is with the National Center for High-Performance Computing, Taiwan, R.O.C. (e-mail: jerry@nchc.org.tw).

Open Science Index, Computer and Information Engineering Vol:7, No:9, 2013 publications.waset.org/16614/pdf

the application. The desktop, data and applications used remain on the remote system. We also implemented a sketch of unified web-based interface to make such a service is simple and easy to use for both casual and expert users in any place and any devices.

The rest of this paper is organized as follows. Section II lists the background. Section III gives a description of software architecture. Section IV gives some details of the implementation. Section V discusses future work and concludes.

## II. Background

### A. Cloud Service Models

Cloud computing enables hardware and software to be delivered as services, where the term service is used to reflect the fact that they are provided on demand and are paid on a usage. There are three service models of Cloud Computing:

1) Infrastructure as a Service (IaaS): it is a provision model in which an organization outsources the equipment used to support associated operations, such as storage, hardware, servers and network components. This service model allows the cloud to operate during high traffic and demanding situations as resources are dynamically increased as they are needed.
2) Platform as a Service (PaaS): it is the delivery of a virtualized application, computing platform and solution stack as a service. It offers a high-level integrated environment to build, test, and deploy custom applications. All applications and infrastructure are run and managed by the services vendor. Generally, developers will need to accept some restrictions on the type of software they can write in exchange for built-in application scalability.
3) Software as a Service (SaaS): it is a software delivery method that provides access to software and its functions remotely as a Web-based service. Providers offer users access to specific application programs controlled and executed on the provider's infrastructure.

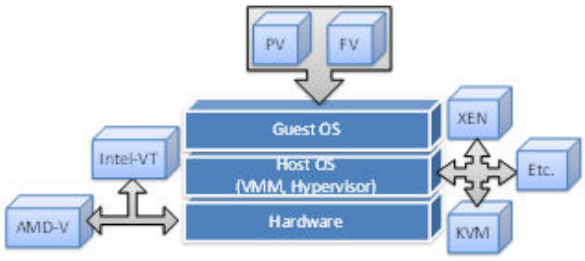



Fig. 1 Virtualization architecture

### B. Virtualization Architecture

Fig. 1 shows the principal architecture of virtualization. Physical hardware resources were divided as virtual resources of virtualization platforms by the monitors, and those virtual resources were assigned to each virtual machine by different application requirements. Furthermore, the monitor provides each Guest OS a set of virtual platform interfaces that constitute virtual machines, acting as a bridge to connect between hardware and virtual devices. Instructions were delivered to hardware layer from virtual platform, which receives results from monitor of virtual machines. Each virtual platform will run independently, although physical resources were shared. By the way, the monitor has two kinds of model, hypervisor and virtual machine monitor (VMM) [14]. The main distinction between Hypervisor and VMM is that the former monitor runs above hardware layer directly with better performance than VMM, such as Xen [15] and VMware's ESX. On the other hand, Microsoft's Virtual PC and VMware's Workstation adopt VMM as monitor of virtual platforms.

For Guest OS, it includes two main virtualization types [16]: para-virtualization (PV) and full-virtualization (FV), which can be both combined with hardware-assisted virtualization. The Guest OS is simulated by modified kernel of Linux with PV, but related devices are not emulated. Instead, all devices are accessed through light-weight virtual drivers offering better system performance and close to the physical machine. But the drawback is that guest kernels must be upgraded to provide new virtual system calls for the new services and all of guest OS must be compatible with the host OS

### C. Virtual Machine

Virtualization technology is able to apply not only to subsystems but to a complete virtual system. To implement a virtual machine, software developers design a software layer to real machines to support the desired architecture. By providing one or more efficient virtual platforms, virtual machines have extended multi-processing systems of the past decade to become multi-environment systems as well. There are many kinds of virtual machine in the market, but not all virtual machines fit to build virtual platforms, so we choose the Kernel-based Virtual Machine to achieve our purpose. KVM is an open source software with GPL, developing by Qumranet company. KVM provide FV solution for Linux on x86 hardware containing virtualization extensions with Intel-VT or AMD-V, and the kernel component of KVM is encased in mainline Linux OS over version 2.6.20, hence the user can set up the virtualization environment of KVM when installing Linux OS conveniently.

Using KVM, we can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware devices, such as network card, disk, graphics adapter, etc. Besides, virtual machine was like a process in queuing system of Linux, and then manager can directly kill any virtual machines by control command.

## III. Architecture

The table I shows the specification information of our cloud service platform named Formosa 3. Formosa 3 [17] is a 64bits high-performance Beowulf cluster located within Southern Business Unit of the National Center for High Performance Computing (NCHC) [18]. It consists of 76 IBM X3550M3 servers as its compute nodes. This self-made cluster was

Open Science Index, Computer and Information Engineering Vol:7, No:9, 2013 publications.waset.org/16614/pdf

designed and constructed by the 'HPC Cluster Group' at NCHC for cloud IaaS service and came online in 2012. Each node has two Six-Core Intel Xeon x5660 2.8GHz processors and 48GB of DDR3 registered ECC SDRAM. All nodes were connected on the InfiniBand high speed network and a private subnet with 1000 Mbits/s Gigabit Ethernet. An additional 4 nodes are used as front ends to interface with cluster, and 4 nodes as storage for the user file systems by Parallel File System.

TABLE I
FORMOSA 3 CLOUD CLUSTER SPECIFICATION

| | |
|---|---|
| CPU | Intel Xeon x5660 six cores 2.8GHz |
| Hard Disk | 80GB SSD |
| Memory | 48GB DDR3 Registered ECC SDRAM |
| Network | 4x QDR 40Gb Infiniband and Gigabit Ethernet |
| Operating System | CentOS 6.3 |
| Hypervisor | Kernel-based Virtual Machine |

The Fig. 2 illustrates the system architecture of our cloud platform. Our system is entirely web-based that way the end user doesn't need to download and install a tool on his computer. In particular, this enables accessing our interface from a wide range of terminals, including mobile devices such as Smartphone or Pad.

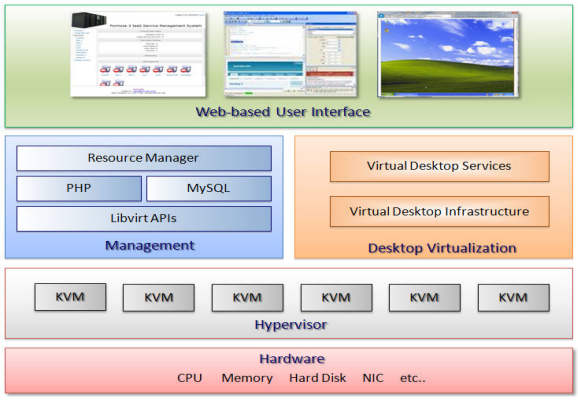



Fig. 2 System architecture

The basic components are as follows:

1) Hardware: there are many physical devices including CPU, memory, hard disk, NIC (Network Interface Card), etc.
2) Hypervisor: we adopt KVM to attain virtualization aim. KVM consists of a loadable kernel module (kvm.ko) that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM also requires a modified QEMU although work is underway to get the required changes upstream. Using KVM, we can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc. The kernel component of KVM is included in mainline Linux, as of 2.6.20.
3) Virtual Desktop Infrastructure (VDI): VDI is a desktop-centric service that hosts users' desktop environment on remote servers, which are accessed over a network using a remote frame buffer protocol.
4) Virtual Desktop Services (VDS): VDS takes full responsibility for hosting and maintaining the compute, storage and access infrastructure, as well as applications and software licenses needed to provide the desktop service.
5) Libvirt APIs: Libvirt [19] is an open source API, daemon and management tool for managing platform virtualization. It can work with a variety of hypervisors in the development of a cloud based solution. Thus, we employ these APIs to control and manage our KVM, and we can switch the underlying hypervisor technology at a later stage with minimal efforts.
6) PHP: we adopt the PHP program language to build all web pages. PHP is an open source server-side scripting language designed for Web development to produce.
7) MySQL: MySQL is the world's most used open source relational database management system (RDBMS) that run as a server providing multi-user access to a number of databases.
8) Resource Manager: for resource management of cloud infrastructure, we developed a resource manager providing control over virtualization requests from user to guarantee the fairness of using the physical machines, priority escalading, and resource partitioning. We also developed a special module called Job Detection Module to detect the actual virtualization job loading. This customized process will calculate the how many physical processors that requests need for finding the physical machine satisfying the given constraints.
9) Web Interface: while our web based interface is built using a Model-View-Controller (MVC) based PHP framework, we also use Python for the command line interface as well. The choice of the Python as the secondary language for the development is supported by the excellent documentation by Libvirt APIs. On the other hand, we employ phpMyAdmin for giving us the ability to interact with our MySQL databases. So we can handy perform maintenance operations on tables, backing up information, and editing things directly.

## IV. IMPLEMENTATION

The virtual desktop and application sharing system allows more than one person to collaborate or develop on a single document in real-time. Currently virtual desktop is delivered using RFB (Remote Frame Buffer) protocol such as Virtual Network Computing (VNC) and Remote Desktop Protocol (RDP). These protocols generally provide methods for accessing a remote virtual desktop or application. This system also presents some novel scenarios for application sharing in single or multiple virtual machines. In a single virtual machine, collaboration scenario can be supported which based on a shared desktop, for example, desktop sharing enables the

instructor and students to work on the same view in a remote teaching system.

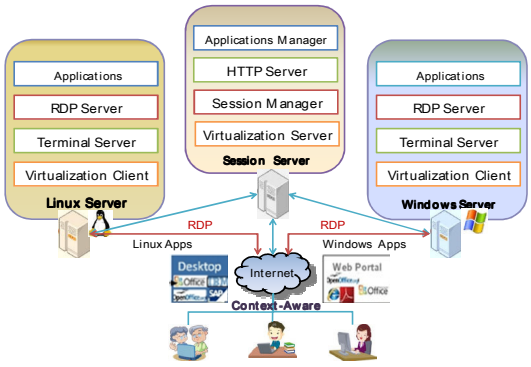



Fig. 3 The virtual desktop and application sharing system

The Fig. 3 shows the virtual desktop and application sharing system architecture. This system has a session server to perform the role of virtualization controller, and to manage user's connection and accounting information. The session server also communicates with application servers to deliver the virtual desktop or application streaming to client devices by RDP technology. Besides, the session server adopts the HTTP server to provide the single web-based portal for user login. It also enhances the system security and elasticity by applying the centralized control of applications and files.

Our system is composed of several modules: Desktop Virtualization Manager, Application Virtualization Manager, User Session Manager and Data Synchronization Manager. The flowchart of software modules is shown in Fig. 4. The following is the details of each module:

- Desktop Virtualization Manager: this module provides virtualization of operating system and customization of remote desktop environment. The development of this module is based on the open virtual desktop software, and designed according to the user requirements. It is responsible for allocating computing resources of cluster server and hypervisors dynamically.
- Application Virtualization Manager: this module provides virtualization of applications and software resources. The development of this module is based on the open virtual application software, and designed according to the user requirements. The Full application virtualization also requires a virtualization layer with the operating system. Application virtualization layers replace part of the runtime environment normally provided by the operating system. The layer intercepts all file and registry operations of virtualized applications and transparently redirects them to a virtualized location.
- User Session Manager: this module is responsible for managing user connection sessions and authenticating accounts information. The user session begins when the user accesses the virtual desktop or application and ends when the user quits the virtual desktop or application from the web browser. It also plays a role of bridge between application servers and client devices. It applied the SSH (Secure Shell) and HTTP protocol to provide the single entry website.
- Data Synchronization Manager: this module is responsible for managing the process of establishing consistency among data from cluster servers to client devices and the continuous harmonization of the data over time. Users and project developers can collaborate or develop on a single file without installing any relative application on their own client device. It also backups and shares the user's data to make the system reliable and elastic.

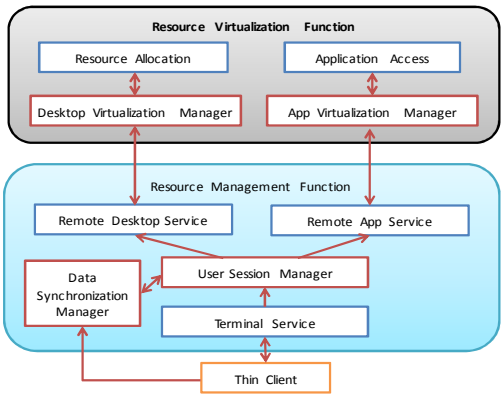



Fig. 4 The flowchart of software modules

To enable collaborations among multiple virtual machines, the application sharing and migration mechanism will be applied. Through presentation streaming redirection and virtual machine cloning technology, an application can be easily shared or migrated. To realize remote application access in the cloud environment, the RFB protocol is used to transfer the virtual desktop of a remote virtual machine. The RFB protocol works at the buffer frame layer and supports the remote access to graphical user interface, and the mouse or keyboard inputs can be transferred to the remote application, thus achieving a transparent access to the application.

While implementing our cloud platform for remote virtual desktop, we came across several issues that have previously not been addressed. For example: for accessing the desktop of the virtual machine users can use Off-the-Shelf tools, ex: VNC and Windows Remote Desktop, etc. Due to the virtual machine may execute on different physical machines every time. This can be troublesome if we provide a fixed public IP address and port for connecting to the user's console of virtual machine. So, we use iptables and thus setup port forwarding connections to the virtual machine that user launched. Our interface will allocate a mapping port dynamically after user creating virtual machine. After that users can connect to the console with the dedicated IP address of the Server and the port which will be forwarded to the appropriate physical machine which is currently hosting the user's virtual machine. On the other hand, our system also supports both multicast and unicast transmissions. For unicast connections, either UDP or TCP can be used. Since TCP provides reliable communication and flow control, it is more

Open Science Index, Computer and Information Engineering Vol:7, No:9, 2013 publications.waset.org/16614/pdf

Open Science Index, Computer and Information Engineering Vol:7, No:9, 2013 publications.waset.org/16614/pdf

suitable for unicast sessions. Multiple TCP clients sharing a single application may have different bandwidths, so an algorithm which sends the updates at the link speed of each client will be developed. For UDP clients, the system controls the transmission rate because UDP does not provide flow and congestion control. Several simultaneous multicast sessions with different transmission rates can be created at the system. The system can share an application to TCP clients, UDP clients, and several multicast addresses in the same sharing session.

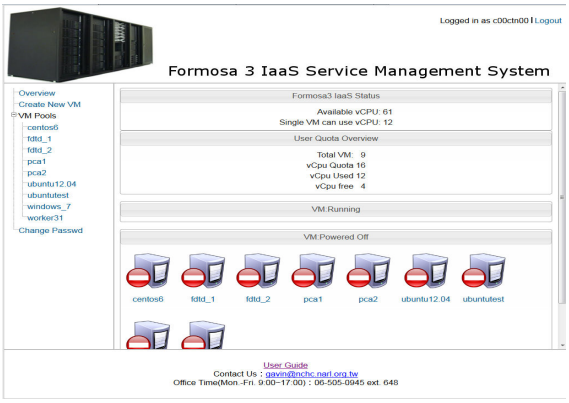



Fig. 5 Cloud service web page

The Fig. 5 shows the web page of our cloud service, we create a test account and login the system to operate virtual machines. Related information and status of virtual machines are showed in this main page, including the quota status of cloud platform and users, users can know that how many cores can be used, how many virtual machines are built, how many cores of user's virtual machines, which network ports are allocated for user's virtual machines. And users can manage their virtual machines including create, delete, shutdown, and add network port.

As shown in Fig. 6, we try to create a virtual machine named TESTVM and open a virtual desktop from cloud service web page. The desktop and application can be streamed from the TESTVM into an isolation virtual environment on the target device. The desktop view is transmitted to the target device and the application execution at run time is controlled by the user session manager and virtualization layer. Users can operate this virtual environment, so it is as well as a standard physical OS environment.

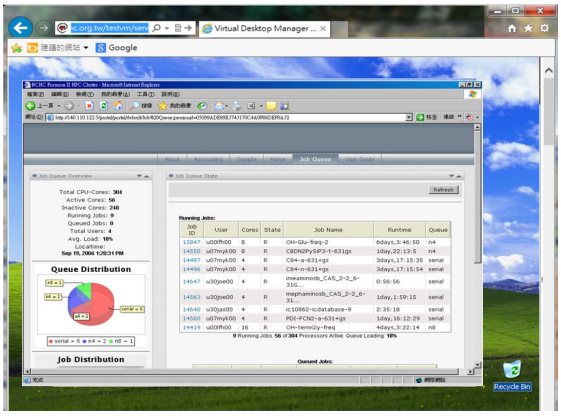



Fig. 6 The terminal of virtual desktop and application

## V. Conclusion

In this paper, we adopt cloud technologies and desktop virtualization to build a virtual desktop and application sharing system which is efficient, resilience and independent of the operating system. This system can provide the application execution takes place on a remote operating system which is linked to the local client device over a network using a remote display protocol. We also implemented a sketch of unified web-based interface to make such a service is simple and easy to use for both casual and expert users. This development has the potential to positively provide an elastic environment for online cloud service, and it is able to help boost user productivity by promoting a flexible model that lets users access their desktop environments from virtually anywhere.

Currently, we support virtual machines that run atop the KVM hypervisor, but plan to add support for Xen, and others in the near future. Also, we plan to optimize the storage performance and adapt power management strategies for physical machines to prevent energy waste in cloud computing platform.

## References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu. "Cloud Computing and Grid Computing 360-Degree Compared," IEEE Grid Computing Environments Workshop, pp. 1-10, 2008.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," 2009.

[3] L. Nussbaum, F. Anhalt, O. Mornard and J.-P. Gelas, "Linux-based virtualization for HPC clusters," Linux Symposium, pp. 221-234, July 2009.

[4] G. Goth, "Virtualization: Old Technology Offers Huge New Potential," IEEE Distributed Systems Online, vol. 8, no. 2, 2007.

[5] R. A. Meyer and L. H. Seawright, "A Virtual Machine Time-Sharing System," IBM Systems Journal, vol. 9, no. 3, 1970.

[6] R. P. Goldberg, "Architecture of Virtual Machines, "National Computer Conference Proceedings, AFIPS Press, vol. 42, pp. 309-318, June 1973.

[7] A. Sultana, B. Daimary, M. Chettri, J. Joseph, "Virtualized Remote Web Desktop," IEEE NCETACS National Conference on Emerging Trends and Applications in Computer Science, 2012.

[8] M. Miller, "Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online," Que Publishing, 2009.

[9] H. Lee, "Design for management software of desktop virtualization solutions," IEEE Information and Communication Technology Convergence, ICTC 2010.

[10] L. Yan, "Development and application of desktop virtualization technology," IEEE Communication Software and Networks ICCSN, 2011.
[11] VMware virtualization, Available at: http://www.vmware.com/
[12] Citrix XenDesktop, Available at: http://www.citrix.com/products/xendesktop/overview.html
[13] KVM - Kernel-based Virtual Machine, Available at: http://www.linux-kvm.org/page/Main_Page
[14] G. J. Popek, and R. P. Goldberg. "Formal Requirements for Virtualizable Third Generation Architectures," Communications of the ACM, vol. 17, pp. 412-421, 1974.
[15] Xen hypervisor, Available at: http://www.xen.org/
[16] W. Chen, H. Lu, L. Shen, Z. Wang, N. Xiao and Dan Chen, "A Novel Hardware Assisted Full Virtualization Technique," The 9th International Conference for Young Computer Scientists, pp. 1292-1297, Nov. 2008.
[17] NCHC Formosa 3 Cloud Cluster, Available at: http://formosa3.nchc.org.tw/
[18] NCHC, National Center for High-performance Computing, Available at: http://www.nchc.org.tw
[19] Libvirt - The virtualization API, Available at: http://libvirt.org/

Open Science Index, Computer and Information Engineering Vol:7, No:9, 2013 publications.waset.org/16614/pdf