World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:11, No:12, 2017

System and Method for Providing Web-Based Remote Application Service

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

***Abstract*—**With the development of virtualization technologies, a new type of service named cloud computing service is produced. Cloud users usually encounter the problem of how to use the virtualized platform easily over the web without requiring the plug-in or installation of special software. The object of this paper is to develop a system and a method enabling process interfacing within an automation scenario for accessing remote application by using the web browser. To meet this challenge, we have devised a web-based interface that system has allowed to shift the GUI application from the traditional local environment to the cloud platform, which is stored on the remote virtual machine. We designed the sketch of web interface following the cloud virtualization concept that sought to enable communication and collaboration among users. We describe the design requirements of remote application technology and present implementation details of the web application and its associated components. We conclude that this effort has the potential to provide an elastic and resilience environment for several application services. Users no longer have to burden the system maintenances and reduce the overall cost of software licenses and hardware. Moreover, this remote application service represents the next step to the mobile workplace, and it lets user to use the remote application virtually from anywhere.

***Keywords*—**Virtualization technology, virtualized platform, web interface, remote application.

I. Introduction

WITH the growing prevalence of virtualization technologies, there is an increasing demand for intellectual information about application virtualization [1]-[3] solutions in the current market. Through application virtualization, almost all applications can be used without installing the application software at a particular environment or changing the local operating system. In other words, the application can be accessed as if it had been installed locally without the need of any modifications to the local client. Resources such as the computing power, data storage and network take care of the execution of these applications. So the display of application delivery over network is a process which has the goal of offering application independent of device and location. Users can work online from anywhere with any device and at any time.

The concept of application virtualization is not new since it has been introduced in the early 2010s. In the early days, many authors [4]-[6] have realized the practicability of desktop virtualization. They adopt platform virtualization in cloud environment to provide remote desktop service. In such cases, platform virtualization acts as a central component that can achieve the purpose of cloud platform and service, and it is an approach to consolidate multiple services into a smaller number of computing resource. A virtualized server allows the computing resources to be shared among multiple isolated platforms which are virtual machines [7], [8]. A virtual machine is a software implementation of a machine that can execute operating system like the physical machine. Each virtual machine has system kernel, operating system, supporting libraries and applications. A hypervisor provides an abstraction of the underlying physical machine, and multiple virtual machines can be operated simultaneously under the hypervisor. This mechanism is able to allow the same virtual machine to be launched on different physical machines. Thus platform virtualization is seen as an enabler for cloud computing, allowing the cloud service provider the necessary flexibility to allocate the cloud resources requested by the user wherever the physical resources are available.

The remote application has received great interests in virtualization research community. Remote application technique has been used in client-server environment, wherein the application programs are stored on a server, and the client computers access the server to obtain functionality from the applications. Many problems will arise in developing remote application; the important one is that the remote application requires specific intelligence about the display characteristics of the client computer. If the environment has many client computers, then the remote application must know the requirements of each client computer. This limit significantly increases complexity of the server software to support the various types of devices. Therefore, it is desirable to develop a unified web-based interface that permits the remote application to operate on many client devices without requiring to have installation of special software or any necessary of configuration.

In this paper, we propose the system and method for adopting the remote application virtualization in cloud platform. We design service-oriented architecture of remote application technology, and present implementation details of the web application and its associated components. We integrate remote application and software streaming technology that make it possible for providing remote application as a service, which is efficient, resilience and independent of the operating system. Our work allows users to launch virtual applications from a remote server that appear on their computer as if it is installed locally, but in reality are running on a remote server. We also implemented a sketch of unified web-based interface to make such a service is simple and easy to use for both beginner and

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

expert in any place and on any device.

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

II. Related Works

We have given many tests on some remote application products before developing remote application as a service. Currently, two well-known solutions for the application virtualization are Microsoft RemoteApp [9] and Citrix XenApp [10]. They have the virtualization interface at the hardware abstraction layer. They virtualize common hardware like processor, memory and storage devices such that multiple operating system instances can be installed on a single machine.

A. Microsoft RemoteApp

Microsoft RemoteApp is a virtual application solution that allows users to run windows-based applications regardless of what operating system they are using. It uses the Remote Desktop Protocol (RDP) [11] to deliver the applications if they are native to the end user's devic. It provides a single infrastructure to enable both virtual and session-based desktops and programs. Currently, users can create a RemoteApp deployment using Microsoft Azure infrastructure directly. The Azure RemoteApp brings the functionality of the on-premises program by remote desktop services. It helps us provide remote access to applications from many different user devices. It basically hosts non-persistent terminal server sessions in the cloud, and we can use them and share them.

B. Citrix XenApp

Citrix XenApp is an application virtualization software that provides Windows applications can be executed on different devices. It provides features such as user self-storage and various user experience enhancements. From [12], a XenApp consists of three parts:

1) A multiuser environment: It allows multiple users to access to Citrix XenApp independently by using Microsoft Windows Server with the Remote Desktop feature.
2) XenApp software: XenApp delivers the remote Windows desktops and applications to local devices via the HDX protocol [13] without the necessity of installing them. HDX protocol is to ensure the XenApp users achieve a high quality virtualization user experience similar to with the desktop computers.
3) Client devices: Citrix XenApp applications can be accessed by various devices using the Citrix Receiver.

Both Microsoft RemoteApp and Citrix XenApp are very close in overall performance overhead. However, Citrix Application streaming overhead was considerably higher than RemoteApp in our tests. Both are commercial products, deployment of this product is not affordable by small enterprises or educational institutions. It has to install some software on user's endpoint devices such as thin client or PCs.

III. Architecture

A. System Architecture

Fig. 1 shows an overview of logical architecture for a standard three-tier architecture. The system is split into multiple modules. In client side, there is a HTML5 web application that supports graphical access directly on the browser and without additional plugins. It rests on today's web protocols: JSON, AJAX, and XML as well. The web application will connect to the Broker. The Broker runs on an Apache server with a servlet container and then reads the specific requests. It is the heart of our system which dynamically loads support for remote application protocols and connects them to back-end platform based on instructions received from the web application. In fact, the Broker is a process which is installed and runs in background, listening for HTTP connections from the web application. The Broker also does not understand any specific protocols, but rather implements just enough of the helper like scheduler and dispatcher to determine which protocol support needs to be loaded and what arguments must be passed to it. Actually, the Broker interprets the contents and handoff the connection to the Proxy then connecting to any number of remote desktop servers on behalf of the user.

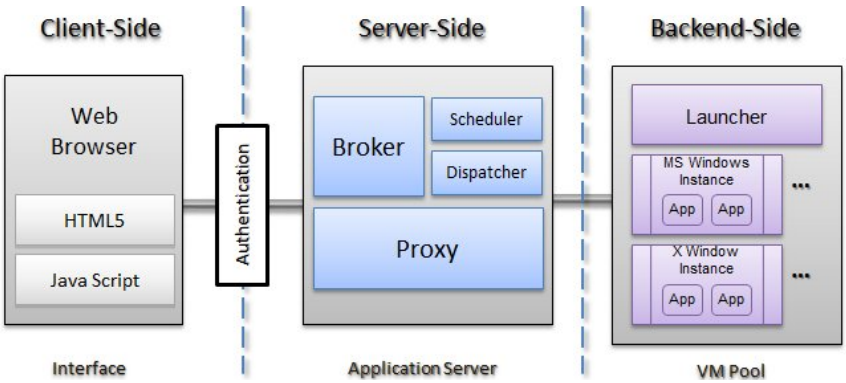

Fig. 1 System architecture

B. Software Architecture

Fig. 2 shows the software architecture of our service system. Our system is entirely web-based that way the end user does not need to install any plugins on the local computer. In particular, this enables accessing the remote application from a wide range of devices, including mobile devices such as Pad or Smartphone.

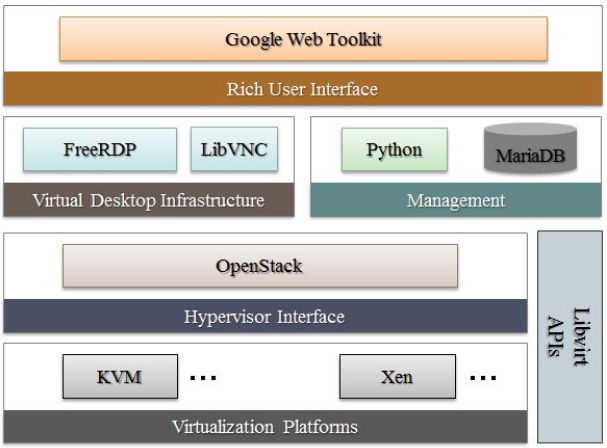

Fig. 2 Software architecture

The following are basic components:

1) KVM: Kernel-based Virtual Machine consists of a loadable kernel module that provides the core virtualization. The goal was to create a virtualization hypervisor that builds on the experience of previous generations of technologies and takes advantage of current hardware. By using KVM, we can build multiple virtual machines running unmodified Linux or Windows images.
2) Xen: It is a lightweight, high performance, open source hypervisor that enables the simultaneous creation, execution and management of multiple virtual machines on one physical machine. It is used as the basis for a number of different open source applications, such as: server virtualization, desktop virtualization, security applications, embedded and hardware appliances.
3) OpenStack: It [14] is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center. It is a collection of open source software that allows us to perform certain functions on the cloud.
4) Libvirt APIs: Libvirt [15] is used to interface with different hypervisors. We use it to interact with the KVM and Xen through a common C programming library.
5) Python: Python is a widely used high-level programming language for general-purpose programming. We adopt the Python program language to build the management web pages.
6) MariaDB: It is an open source relational database management system based on MySQL. We use MariaDB to provide the database service for web applications hosted with application server.
7) Google Web Toolkit [16]: Our web pages were built by using the Model-View-Controller (MVC) based Google Web Toolkit framework. The SDK provides a set of core Java APIs and Widgets. These allow us to write AJAX applications in Java and then compile the source to highly optimized JavaScript that runs across all browsers.

C. Service Platform

Table I is the specification information of our service platform named Formosa 5 [17]. Formosa 5 is a high-performance Beowulf cluster located at National Center for High Performance Computing (NCHC) [18]. It consists of 84 servers as its compute nodes. This self-made cluster was constructed by the 'HPC Cluster Group' at NCHC for cloud services and came online in 2012. Each machine has two six cores Intel Xeon x5670 2.93GHz processors and 96GB of DDR3 registered ECC SDRAM. All nodes were connected on InfiniBand network and a private subnet with Gigabit Ethernet.

TABLE I
FORMOSA 5 CLOUD CLUSTER SPECIFICATION

CPU	Intel Xeon x5670 six cores 2.93GHz
Hard Disk	120GB SSD
Memory	96GB DDR3 Registered ECC SDRAM
Network	4x QDR 40Gb Infiniband and Gigabit Ethernet
Operating System	CentOS 6.4
Hypervisor	Kernel-based Virtual Machine

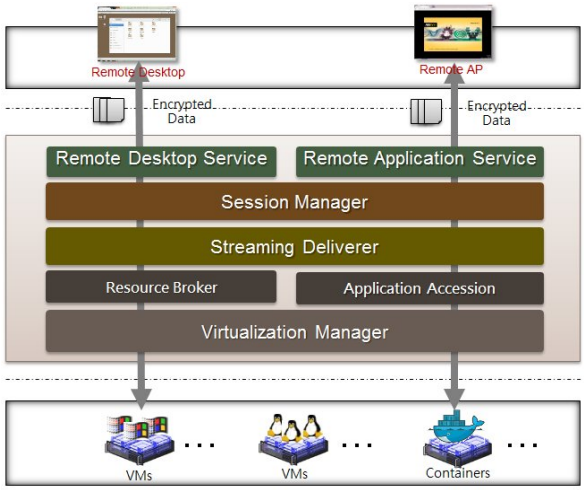

Fig. 3 System module

IV. IMPLEMENTATION

This section aims to explain the details of implementation involved with related modules. As shown in Fig. 3, these modules held in system can be fully integrated with the client-side portal and fully combination in between the underlying virtualization platform. The system modules are as follows:

A. Resource Broker

A resource broker determines which VM-hosted remote applications will be available to a user and which user will get access to which applications, both persistent and non-persistent. When using a VM-hosted virtual application infrastructure for

this, it is possible to either designate dedicated applications or a pool of remote applications. And then the application accession module can automatically create, remove or pause remote applications. Depending on the virtualization platform, the application accession may have additional functions, such as a web interface that can create secure connections to remote applications, directory services and integration with remote agent. Depending on the requests, it is possible to execute remote applications centrally on a terminal program or on web page.

B. Session Manager

With session manager every user has his or her unique connection session. Such sessions share the same server operating system for each user, as opposed to VM-hosted infrastructure where each user has access to their own dedicated remote application. Session manager is a solution for the remote access to desktops and applications that are run on an application server. Access to the application is not tied to the end-user machine, and programs are executed on the server centrally.

C. Streaming Deliverer

The delivery process of transporting the application specific resources or data to the end-point at the time the application is executed is called streaming deliverer. The application is running and only the minimum amount of data is delivered to the client before application is launched. This result in the first time application launch for the user, it also makes it possible to keep user images and results in reduced load on the network. Additional functions of the application are delivered without user intervention. The packages of application are stored on a centralized server, which can be a shared or dedicated infrastructure platform. The streaming deliverer transports the data over network in an efficient, secure and optimized way.

D. Virtualization Manager

This module provides virtualization of operating system and customization of remote application environment and software resources. The development of this module is designed according to the user requirement, and based on the open virtual desktop software. The full application virtualization also requires a virtualization layer with the operating system. The layer intercepts all file and registry operations of virtualized applications and transparently redirects them to a virtualized location.

Our web-based remote application system is built based on the VDI (Virtual Desktop Infrastructure) framework [19]. VDI is a remote desktop solution for providing remote access to Windows desktops. The implementation of VDIs means that applications are no longer bound to a location or end-user appliance. So in our system, each user can have their own unique, personalized, fully independent workplace. The information is sent to the client screen via a remote display protocol such as RDP and VNC. These protocols used for displaying the correct information depend on the operating system and the type of application. For Windows platform, we implemented a RDP interpreter based on FreeRDP [20] library to capture the graphical content and identify the basic semantic information. This semantic information is described by using the RDP specification. For Linux-based service platform, we implemented a VNC interpreter based on the LibVNC [21] library. These protocols generally provide methods for accessing the remote applications. This unique functionality also gives administrator complete control on how applications are delivered and interact with client devices. As with other application delivery solutions, our service platform consists of various infrastructure components that facilitate provisioning, application delivery, load balancing, session control and secure access to virtual machines.

Fig. 4 illustrates a simple web interface for remote WordPad application hosted on Windows 7. WordPad is a basic text-editing program available in the Microsoft Windows operating system. We can use WordPad to create documents such as letters, notes and posters on remote environment through a web-based interface. Although the program is running on a remote computer, they behave as if they are running on your local computer. This feature enables accessing the interface natively on different mobile devices particularly.

Fig. 5 is web-based SSH terminal program running on remote Windows 7 system. SSH is a text protocol, the implementation of this interface is actually a combination of a terminal emulator and SSH client, because the SSH protocol isn't inherently graphical. So this widget emulates a terminal on the server side and draws the screen of this terminal remotely on the client.

V. Conclusion

Client-side application virtualization reduces the cost of installing, testing and supporting variable applications. Using isolation and application streaming technologies, server-side application virtualization enables local virtualized applications. Rather than installing applications on each user's computer, the remote applications are streamed to a protected isolation environment on their client device.

In this paper, we presented the system and method for embracing the remote application virtualization in cloud platform. We developed a service-oriented architecture of remote application technology, and gave the details of the web application and its associated modules. We integrated resource scheduling and allocation that make it possible for providing remote application as a service, which is efficient, resilience and independent of the operating system. This system allows users to launch remote applications from the back-end server that appear on their device as if it is installed locally. We also implemented a unified web-based interface to make such a service is easy to use for users in any place and on any device.

Currently, this service normally enforces authentication, requiring all users to have a corresponding set of credentials. An important next step involves online model refinement to address dynamically changing application behavior. It would be interesting to see if refining the schema to incorporate the influence of additional underlying hardware, especially resource allocation and data storage. We also plan to reduce the communication overhead of connection proxy and import some

smart management strategies for physical machines to reduce power consumption in service platform.

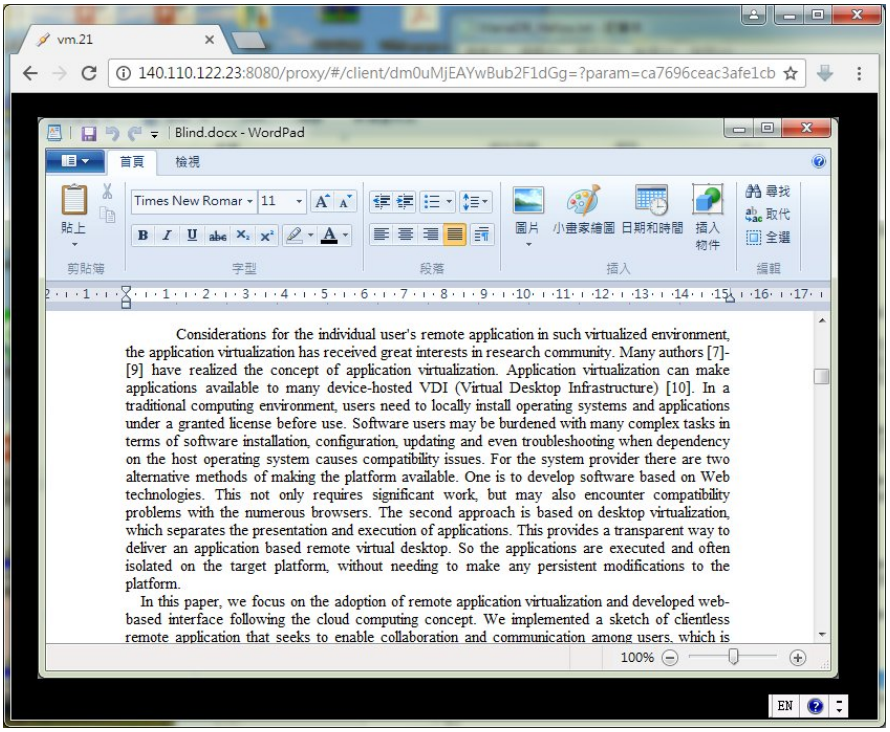

Fig. 4 Web-based remote WordPad on Windows 7

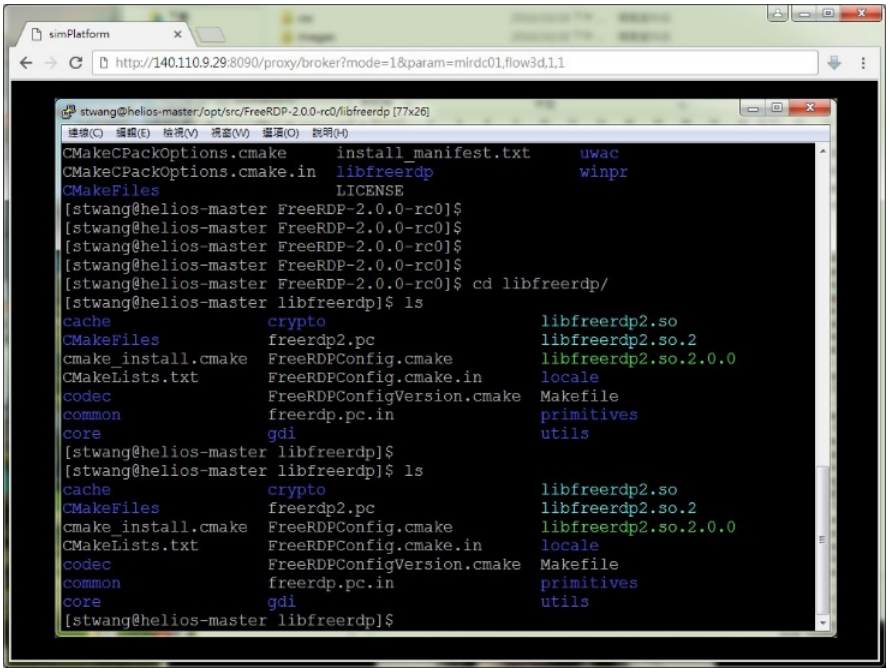

Fig. 5 Web-based remote SSH terminal on Windows 7

REFERENCES

[1] Datta, Anindya. "Method and apparatus for performing application virtualization." U.S. Patent Application No. 10/837,247.
[2] Yan, Li. "Development and application of desktop virtualization technology." Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on. IEEE, 2011.
[3] Tang, Chang Bin, and Fen Zhou. "Generalized application virtualization method for business use on the web and the mini server using this method." U.S. Patent Application No. 11/830,493.
[4] Lai, Guangda, Hua Song, and Xiaola Lin. "A service based lightweight desktop virtualization system." Service sciences (ICSS), 2010 international conference on. IEEE, 2010.
[5] Jang, Su Min, Won Hyuk Choi, and Won Young Kim. "Client rendering

method for desktop virtualization services." ETRI Journal 35.2 (2013): 348-351.
[6] Lee, HyungJik, and JeunWoo Lee. "Design for management software of desktop virtualization solutions." Information and Communication Technology Convergence (ICTC), 2010 International Conference on. IEEE, 2010.
[7] R. A. Meyer and L. H. Seawright, "A Virtual Machine Time-Sharing System," IBM Systems Journal, vol. 9, no. 3, 1970.
[8] R. P. Goldberg, "Architecture of Virtual Machines, "National Computer Conference Proceedings, AFIPS Press, vol. 42, pp. 309-318, June 1973.
[9] Microsoft RemoteApp, http://technet.microsoft.com/en-us/library/cc755055.aspx.
[10] Citrix XenDesktop, http://www.citrix.com/xenapp, 22/09/2017.
[11] Remote Desktop Protocol (RDP) Features and Performance, Microsoft Corporation, Jun. 2000.
[12] Cvetanov, Konstantin. Getting Started with Citrix XenApp 7.6. Packt Publishing Ltd, 2015.
[13] HDX technologies for optimizing application and desktop delivery, https://www.citrix.com/products/xenapp-xendesktop/hdx-technologies.html, 22/09/2017.
[14] Sefraoui, Omar, Mohammed Aissaoui, and Mohsine Eleuldj. "OpenStack: toward an open-source solution for cloud computing." International Journal of Computer Applications 55.3 (2012).
[15] Libvirt: The virtualization API, https://libvirt.org/, 22/09/2017.
[16] Wargolet, Steve. Google Web Toolkit. Technical report 12. University of Wisconsin-Platterville Department of Computer Science and Software Engineering, 2011.
[17] NCHC Formosa 5 Cloud Cluster. http://formosa5.nchc.org.tw/
[18] NCHC, National Center for High-performance Computing. http://www.nchc.org.tw/, 22/09/2017.
[19] Velte, Anthony, and Toby Velte. Microsoft virtualization with Hyper-V. McGraw-Hill, Inc., 2009.
[20] FreeRDP - a free remote desktop protocol client. http://www.freerdp.com/, 22/09/2017.
[21] LibVNC, https://github.com/LibVNC, 22/09/2017.