International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:6, 2016

An Integrated Cloud Service of Application Delivery in Virtualized Environments

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

***Abstract*—**Virtualization technologies are experiencing a renewed interest as a way to improve system reliability, and availability, reduce costs, and provide flexibility. This paper presents the development on leverage existing cloud infrastructure and virtualization tools. We adopted some virtualization technologies which improve portability, manageability and compatibility of applications by encapsulating them from the underlying operating system on which they are executed. Given the development of application virtualization, it allows shifting the user's applications from the traditional PC environment to the virtualized environment, which is stored on a remote virtual machine rather than locally. 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 platform maintenances and drastically reduces the overall cost of hardware and software licenses. Moreover, this flexible and web-based application virtualization service represents the next significant step to the mobile workplace, and it lets user executes their applications from virtually anywhere.

***Keywords*—**Cloud service, application virtualization, virtual machine, elastic environment.

I. Introduction

VIRTUALIZATION technology [1], [2] 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 [3], [4]. 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 application, application virtualization has received great interest in the virtualization research community. References [5]-[7] have realized the concept of application virtualization. Virtualized applications eliminate nearly all of the complexities and support issues associated with delivering and accessing traditional applications for thin-client deployments. We realized this impact and also tried many implementations before integrating our virtual application service. Citrix XenApp [8] is an end-to-end Windows application delivery system that offers client side application virtualization. All applications are managed in a centralized controller, but are streamed to the user's machine and run in an isolation environment. Even the centralized administration can deploy the applications to the end user transparently as a service and their usage can be tracked and monitored. However, for existing Citrix users, Citrix has a lower total cost of ownership due to the overlap with the existing Citrix Access Infrastructure, and applications which include the installation of a service cannot be virtualized. Microsoft Application Virtualization [9] is the application virtualization solution from Microsoft. It is composed mainly of two components - System Guard and SoftGrid Sequencer. System Guard tracks configuration repositories and resources used by the application and redirecting them to the virtualized instances of the resources. SoftGrid sequencer is the server side component which packages an application for virtualization and streaming. The main advantage is sequencer uses special approach, which ensures that only the application code needed is transferred. As a result, applications will start up very quickly. But the disadvantage is applications needing product activation that use hardware characteristics cannot be virtualized. Moreover, when the application is started on another machine, the hardware characteristics will have been changed.

In this paper, we aim at the adoption of application virtualization and cloud technologies. We integrated a virtual 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 delivered to the end user virtually by passing only screen pixels, keystrokes and mouse actions between the client and server over the network. The data and applications used remain on the remote system. We also implemented a sketch of

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).

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.

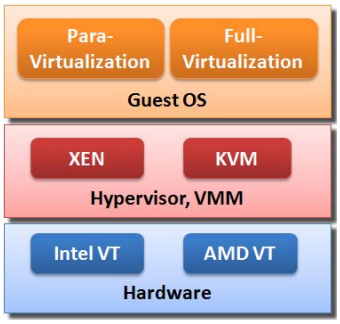

Fig. 1 Virtualization architecture

II. Background

A. 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. The monitor module has two kinds of models: hypervisor and virtual machine monitor (VMM) [10]. 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 [11] and VMware's ESX [12]. 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 [13]: 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

B. 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 can be built as a virtual platform, so we choose the Kernel-based Virtual Machine [14] 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.

III. Architecture

Table I shows the specification information of our cloud service platform named Formosa 3. Formosa 3 [15] is a 64bits high-performance Beowulf cluster located within Southern Business Unit of the National Center for High Performance Computing (NCHC) [16]. It consists of 76 IBM X3550M3 servers as its compute nodes. This self-made cluster was 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

Fig. 2 illustrates the system architecture of our cloud platform. Our system is entirely web-based that way the end user does not 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.

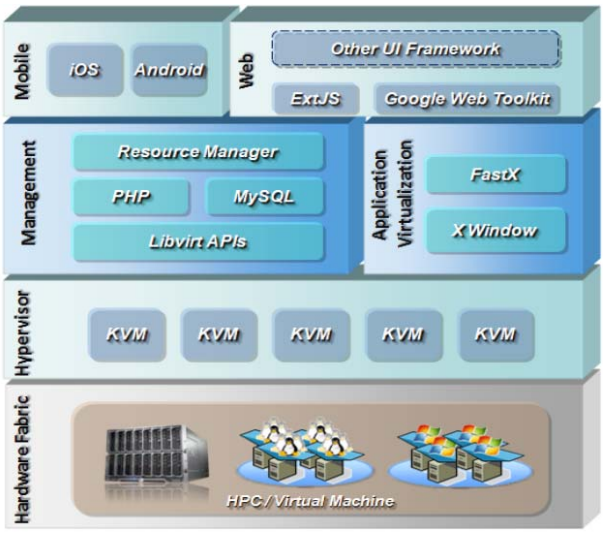

Fig. 2 System architecture

The basic components are as follows:

1) Hardware Fabric: there are many physical facilities including High Performance Cluster, Linux Virtual Machines, 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) Libvirt APIs: Libvirt [17] 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.
4) MySQL: MySQL [18] 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.
5) 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.
6) X Window: X Window is a windowing system for graphics workstations developed at MIT. It is based on a client/server model. A networked computer runs an X server, and client programs running on connected workstations request services from the server. The server handles input and output devices and generates the graphical displays used by the clients.
7) Google Web Toolkit [19]: GWT is a development toolkit for building and optimizing complex browser-based applications. The GWT 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.
8) Web Interface: while our web based interface is built using a Model-View-Controller (MVC) based PHP framework, Python was adopted as the command line interface in this platform. 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. Development

Virtualized application is a process which has the goal of offering applications independent of location and device, so users can work online and anywhere with any device and at any time. When we were studying and determining which application virtualization solution suits with our virtualized environments. We realized that the performance of application streaming is a key issue. Application virtualization is often combined with application streaming, a technology that streams portions of the virtualized application to users' computers as needed. With application streaming, the virtualized application image is stored on a server that client software on users' machines connects to. Instead of the entire file being transferred, the client requests portions of the

application only as needed, speeding up the time it takes to launch the program.

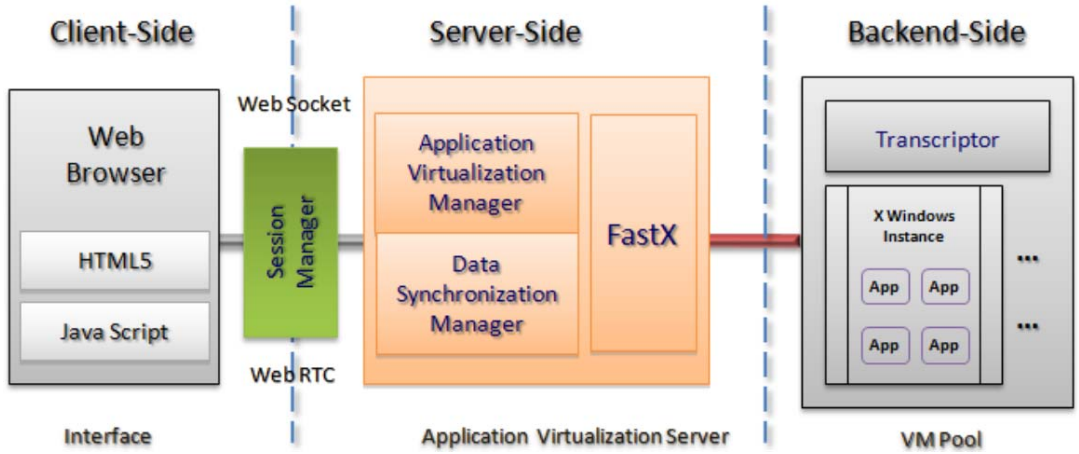

Fig. 3 The application delivery system

Fig. 3 provides an overview of the application delivery architecture for a standard, three tiers development. The framework architecture is composed of three main parts: the Client-Side, the Server-Side and Backend-Side. The internal components of every architecture part and their provided functionalities are discussed in the following:

1) Client: the Client is a web-based GUI allowing users to operate remote application. Currently we adopt GWT to allow the use of existing Java knowledge and tools to build high performance, web applications. GWT abstracts away many complexities of web application development by not requiring us to learn Javascript and HTML. It rests on today's web standards: AJAX, JSON and HTML5 [20] as well.
2) 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 HTTPs protocol to provide the single entry website.
3) 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.
4) 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.
5) FastX: FastX [21] is a simple and affordable X Windows terminal emulator which displays both remote Linux/Unix desktops and individual application. Now we use the trial version of FastX to test and verify seamless access to X Windows and X applications. We integrated FastX with the managers to transparently provide secure encrypted communications by using SSH connection protocols, and let the X application can be launched from within a Web browser, or simply by clicking on an icon on the user's desktop in our platform.

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 streamed or migrated. To speed up the performance of remote application access in the cloud environment, WebSocket [22] protocol is used to transfer the virtualized application of a remote virtual machine. The WebSocket 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 and real time communication in the browser.

While integrating these different modules for remote virtualized application service, we came across several issues that have previously not been addressed. For example: for accessing 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 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.

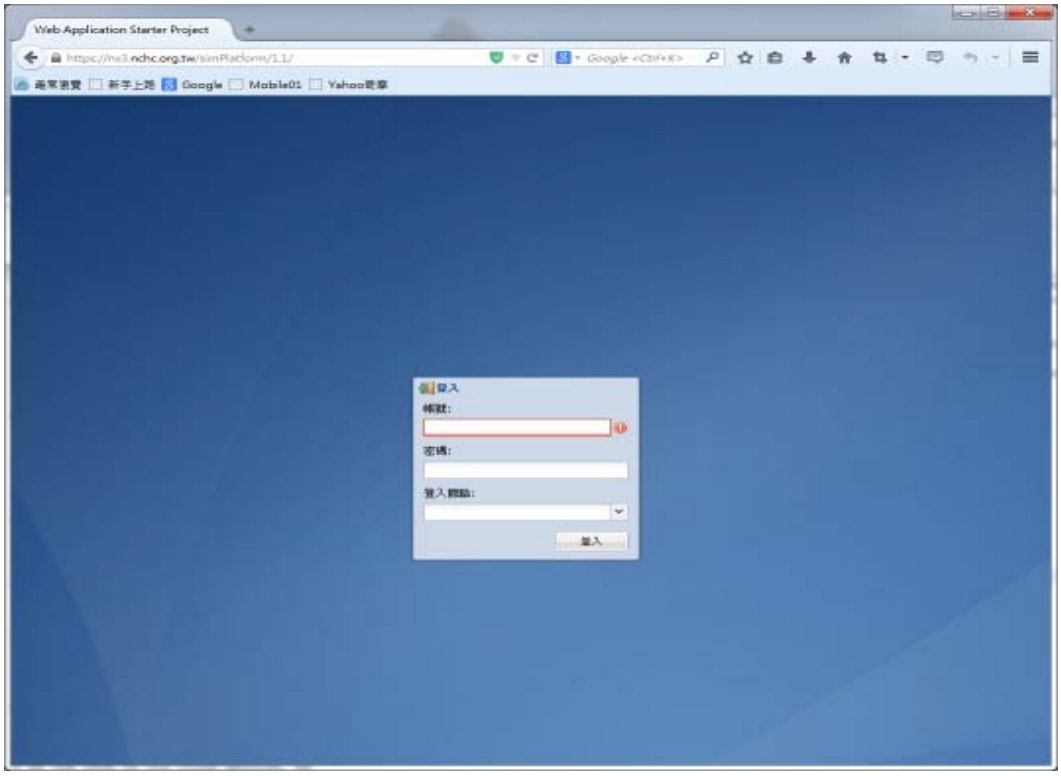

Fig. 4 The cloud platform web portal

Fig. 4 shows the web portal of our cloud platform, once user has logged in he/she should be able to choose which widget wants to use.

Fig. 5 illustrates the X-Window terminal widget, the main task of this widget is to allow user to launch remote xterm application and access to back-end servers through a web-based interface. This feature also allows the other server which installed different operating system to be accessed natively from a mobile device such as a tablet.

Unlike terminal widget, Fig. 6 is web-based text editor widget. It launches remote xedit application. Xedit is an X-Window editor specifically designed for writing programs, but it's also useful for writing any kind of text. It works well in web browser.

V. Conclusion

In this paper, we aim at the adoption of application virtualization and cloud technologies. We integrated a virtual 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 delivered to the end user virtually by passing only screen pixels, keystrokes and mouse actions between the client and server over the network.

Currently, one of the most important things when this service comes to using the virtualized applications is performance. It's about user-acceptance. So we will plan to add communication method with data compression and encryption for mobile computing environments in the near future. There is also a plan to optimize the storage performance and adapt power management strategies for physical machines to prevent energy waste in cloud platform.

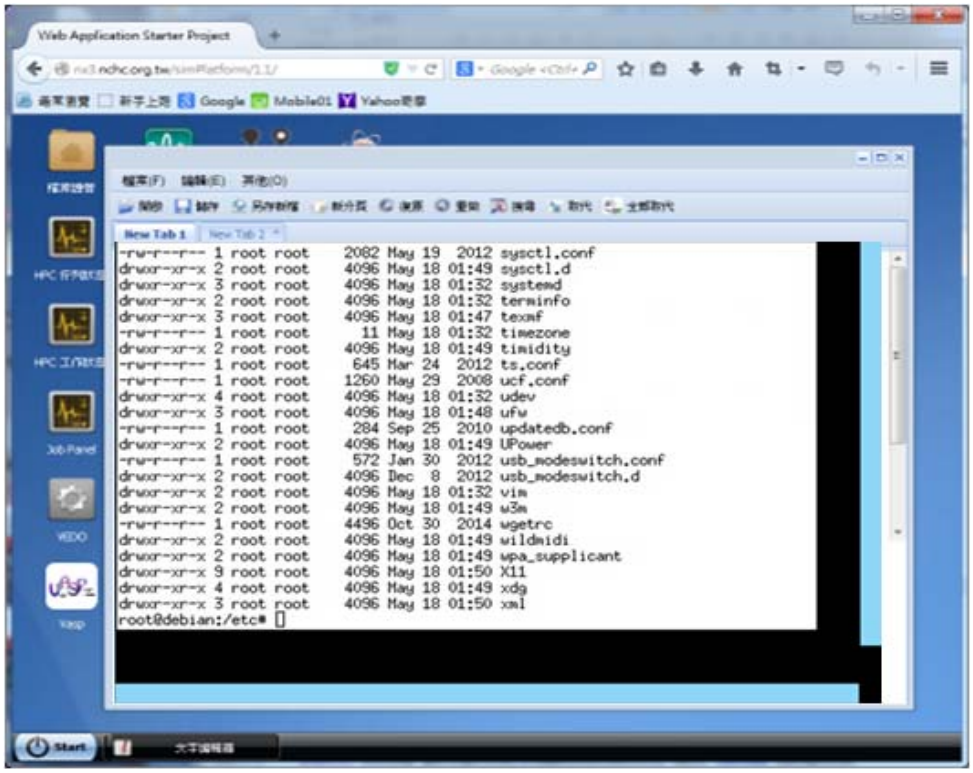

Fig. 5 Web-based remote 'xterm' application

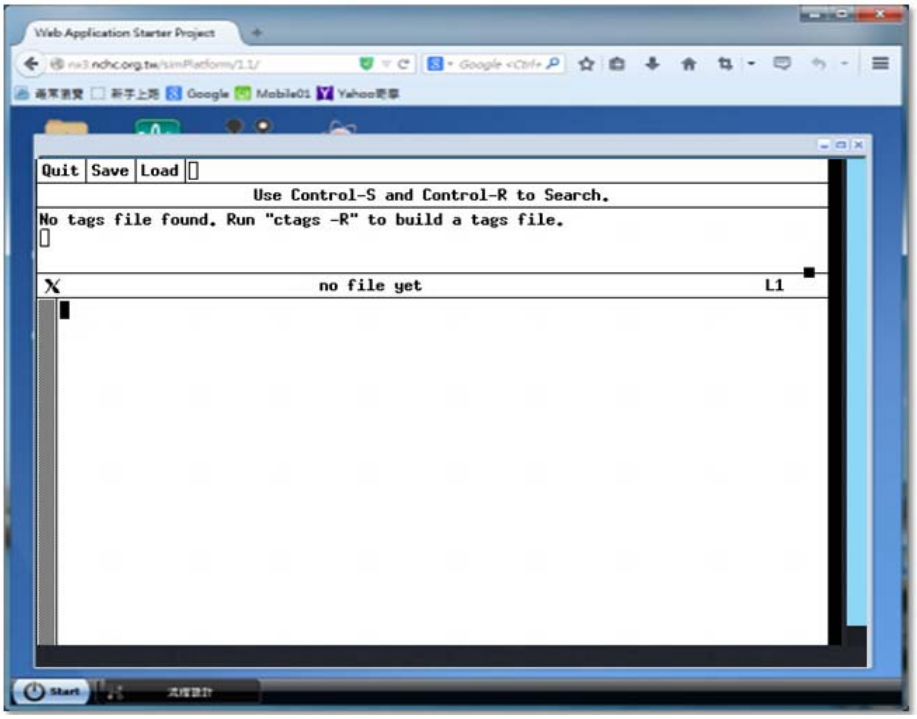

Fig. 6 Web-based remote 'xedit' application

REFERENCES

[1] Nussbaum, Lucas, et al. "Linux-based virtualization for HPC clusters." Montreal Linux Symposium. 2009.
[2] Goth, Greg. "Virtualization: Old technology offers huge new potential." IEEE Distributed Systems Online 2 (2007): 3.
[3] Meyer, Richard A., and Love H. Seawright. "A virtual machine time-sharing system." IBM Systems Journal 9.3 (1970): 199-218.
[4] Goldberg, Robert P. "Architecture of virtual machines." Proceedings of the workshop on virtual computer systems. ACM, 1973.
[5] Yan, Li. "Development and application of desktop virtualization technology." Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on. IEEE, 2011.
[6] Ganji, Rama Rao, et al. "HTML5 as an application virtualization tool." Consumer Electronics (ISCE), 2012 IEEE 16th International Symposium on. IEEE, 2012.
[7] Chiueh, Susanta Nanda Tzi-cker, and Stony Brook. "A survey on virtualization technologies." RPE Report (2005): 1-42.
[8] Musumeci, Guillermo. Getting Started with Citrix XenApp 6. Packt Publishing Ltd, 2011.
[9] Microsoft Application Virtualization, Available at: https://technet.microsoft.com/en-us/windows/hh826068.aspx
[10] Popek, Gerald J., and Robert P. Goldberg. "Formal requirements for virtualizable third generation architectures." Communications of the ACM 17.7 (1974): 412-421.
[11] Xen hypervisor, Available at: http://www.xen.org/
[12] VMware virtualization, Available at: http://www.vmware.com/
[13] Chen, Wei, et al. "A novel hardware assisted full virtualization technique." Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for. IEEE, 2008.
[14] Kivity, Avi, et al. "kvm: the Linux virtual machine monitor." Proceedings of the Linux Symposium. Vol. 1. 2007.
[15] NCHC Formosa 3 Cloud Cluster, Available at: http://formosa3.nchc.org.tw/
[16] NCHC, National Center for High-performance Computing, Available at: http://www.nchc.org.tw
[17] Libvirt - The virtualization API, Available at: http://libvirt.org/
[18] DuBois, Paul. MySQL: the definitive guide to using, programming, and administering MySQL 4. Sams, 2003.
[19] Johnson, Bruce, and Joel Webber. Google web toolkit. Addison-Wesley, 2007.
[20] Hickson, Ian, and David Hyatt. "Html5." W3C Working Draft WD-html5-20110525, May (2011).
[21] FastX, Available at: https://www.starnet.com/fastx/
[22] Wang, Vanessa, Frank Salim, and Peter Moskovits. The definitive guide to HTML5 WebSocket. Vol. 1. New York: Apress, 2013.