International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.4 (2014), pp.85-92
http://dx.doi.org/10.14257/ijmue.2014.9.4.09

An Extension of Virtual Computing Laboratories: Scalable Virtual Desktop Infrastructure on Cloud Computing

Dongho Song[1], Ji-Ae Shin[2] and Yeonjin In[2]

[1] *Dept. Of Computer Engineering, Korea Aerospace Univ., 76 Hanggongdaehang-ro, Deogyang-gu, Kgoyang-City, Gyeonggi-do 412-791, Korea*
[2] *SoftonNet Inc., 6F, Goodman Tower, 689-2, Yeoksam-dong, Kangnam-gu, Seoul, 135-080, Korea*

dhsong@kau.ac.kr, {jiae.shin, yjin}@softonnet.com

Abstract

Virtual Desktop Infrastructure (VDI) based on virtual computing laboratories (VCL) model has been implemented in a small or medium scale on cloud computing in universities. Conventional physical computing laboratories with thousands of PCs are still widely used and co-exist with the VCL, however, the conventional PCs are not fully utilized in VCL. The demand of VDI is growing but the numbers of VDIs are bounded by the capacities of host servers. To have scale-up VDI, system feasibility in terms of functionality and performance and scalability of virtual machines should be tested. We propose an extended model of VCL that can scale-up VDIs by running on an integrated virtual and physical lab environment. End-users can build their own workspaces on a host server and migrate to PCs in physical lab. The paper concludes with discussions on the benefits, problems, and performances of the extended model of the VCL.

Keywords: *Computing laboratory, E-learning, Virtual lab, Scalable VDI*

1. Introduction

In order to extend university computer labs to the outside of campus anytime anywhere [1], we propose a cloud computing model. There is previous work such as NCSU VCL model in which the goal is partially achieved with some unresolved issues [3, 6]. VDI is good for remote access for limited users who got ids of the service, but it is bounded by the capacity of host server machines on a cloud computing. VDI solutions are still cost high to apply for a whole campus; in the conventional VCL model, it has not addressed how the VCL models can utilize campus-wide distributed PC resources.

Students yearn not only to use various PCs in campus, but also to use own laptop or PCs at homes or in public places to do seamlessly work from the VCL at their university. It can be even more effective if the smart devices that students use are able to be serviced by being connected to the computer labs. In the NCSU VCL model mentioned above, students can only work by connecting to the VDI on the host server. If the VCL is full, they cannot continue to work at home PCs as they worked at lab PCs, because the VMs cannot come out from host servers. In case of just migrating content files, such as .c, from a VM on the host to the local PCs, a compiler or various applications used on the host may not be installed on the local PC as well as the profiles may not be configured in the same way as Virtual Machine(VM) on the host. Considering that over 95% of students have the privilege of having smart phones and accessing to the Internet, it is a pity that at Virtual Computing Lab(VCL), their devices and

ISSN: 1975-0080 IJMUE

networks are not always connected spatially and chronically with their VMs on the host at VCL.

Therefore, we are motivated to extend the central host server based on VCL model to include and utilize PCs in physical PC labs, public PCs on the campus and even PCs at home. Three different ways can be proposed: (1) migrating workspace from a VM on the central host to local PCs in campus or at home; (2) migrating workspace from a VM on the local PC to other VM on other PC; (3) connecting any VMs from the end-users' smart devices. The key idea is migration of workspaces of users from one VM to another VM on running on the central host or the local PCs, and vice versa. This enables the end-users to have seamless working with the same workspace on VDI and on their PCs at the physical lab or even on PCs at home. The workspace is defined as a working image of computer desktop, that consists of OS images, applications, storages, and furthermore user's personal profiles, and can be executed, migrated, and stored on one VM to another VM.

Under the free workspace migration distributed computing environment, a scale-up virtual computing laboratory can be considered by extending the number of VDIs which is not just bounded by the power of host server machines but by running on any PCs with client hypervisor installed in the university campus. In section 2, background works are discussed. and section 3 proposes an extended VCL model and implementation. Section 4 describes on the experimental data and analysis of the results followed by the conclusion at the last section.

2. Backgrounds

In Burd [2, 3], Wenhong [9], and Vouk [10], virtual computing lab is discussed and virtual lab and physical lab are compared. However, their basic reference model is host based VDI and do not include the migration of a whole users' workspace from a central VDI to other VDI on the local PCs in campus. Thus, the difference is as follows: the previous models are host based VDI; on the other hand, our model is integrated host based VDI and client PC based VDI, and VMs with a whole workspace are being provisioned.

Song [7] proposes how to extend a VCL model by running VDIs on client PCs with client hypervisor as well as how to connect to the VDIs. However, scalability issue of the whole VDIs in the VCL system is not considered in the work [7].

3. Scale-up VDI Modeling

3.1. A System and Operational Service Model

A system model of workspaces with VDIs that comprise of in-campus and out-campus is shown in Figure 1; software configuration and operational service model are shown in Figure 2. The hardware and software layers in host machines and PCs are basically the same in terms of hypervisor, VMs, connection brokers, and screen transfer protocols. In other words, every hardware layers are supposed as heterogeneous for hosts and PCs. On the top of that, the hosts or PC virtualization layers, called hypervisor, are installed; then Windows 7 or Linux are installed as the OS you need; then the Apps you need can get from application virtualization server through streaming.

To create a workspace, for instance, you will install Window OS and MS Visual Studio, and take a copy of a master image of everything as one. Then this master image can be used as a workspace for you and it can be executed in university lab PCs, and public PCs in the libraries. Also, it can be performed in home PCs outside the university campus or in PCs in cafes, which is the gist of this model.

Here the key of the scale-up system model [1, 5, 8] is in the central VDI management: VDIs running on the host servers as well as on the client PCs are all centrally managed by a single VDI manager. This allows scalable VDIs over the boundary of host servers in a central cloud computing center to PCs in the physical lab. Notice that the VDI is a proprietary system developed by SoftonNet Inc.

3.2. Implementation of the System Model

Virtual computing lab consists of the server side and the client side. On the server side, there are a single Z!Desktop VDI managing server and multiple operational servers, Z!Sync server, Z!Stream application virtualization server, Z!BootOS server, and Z!Storage software defined storage system. These servers are all running on the hosts in a rack. XEN hypervisor are running on X86 machines for VDI host server, and on top of that, a host linux (Dom0) is running and guest VMs (DomU) such as Windows7 or Linux are running. These guest VMs communicate to remote PC terminals or iPads through remote desktop protocols (RDP). On the client side, there are many PCs everywhere, and smart devices in student's hands anywhere in or outside of campus. Assume that the client PCs in physical labs are all running client hypervisor (Xenclient) locally on bare-metal hardware, Linux kernel as a host, and Windows or Linux as guest VMs. Note that users can connect a monitor to the PC's graphic card, because the local client hypervisor can drive the graphic card directly, which means that it is not necessary to use RDP for a local monitor. Z!Sync is a management tool to manage the images for the local client hypervisor and those stored in server and to synchronize them. Z!Storage is a software defined storage with NAS structure. VM master images for each individual student, each classes are stored centrally on Z!Storage server. It can be used for booting remote PCs at physical lab, VMs for VDI at cloud server, and VMs for local client hypervisor at home PC.

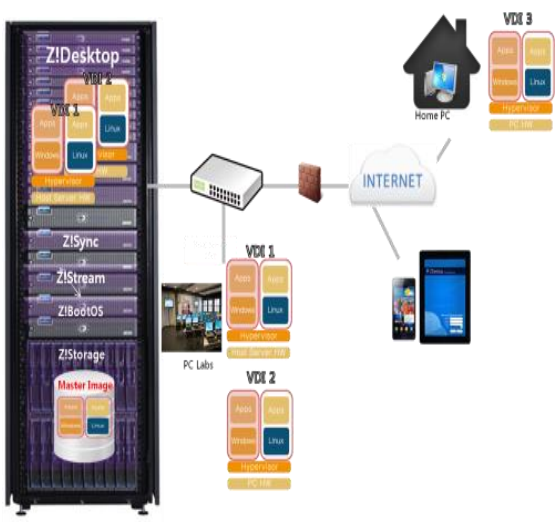

Figure 1. A system model for workspaces

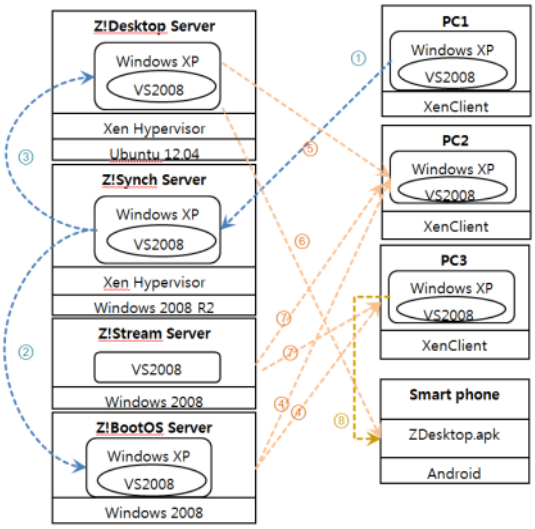

Figure 2. An operational service model

3.3. Operation Scenarios

Based on the model described earlier, general working scenario can be divided into two phases: an administration phase and a user phase. Administration phase is depicted in blue dotted arrow lines in Figure 2 and user phase is in red dotted line.

A. When a new semester starts, a teaching assistant for the class creates an initial workspace for a student by preparing a clean PC1, installing required OS and applications, taking copy of the image as a workspace, and registering it to Z!Sync (①).

B. Z!Sync store it inside Z!Storage, register it to Z!BootOS and ready for release (②).

C. When the student login to a VCL website with his ID on his iPhone or on his PC2, it looks for a VM available on Z!Desktop host server, first. If a VM is found on Z!Desktop, then it connects to it with his iPhone. Z!Sync brings the student's VM image and boots the VM(③). Now, RDP connection is made between his iPhone and the VM (⑥).

D. If there are no available VM on Z!Desktop hosts, then Z!Desktop central managing server looks for other available VM on PCs in physical lab such as PC1, PC2, or PC3 (④ , ⑦). and connects to it with his iPhone (⑧). This is the scale-up VDI process. The rest of the processing is the same as that on hosts mentioned above.

E. Once Z!Desktop central managing server selects a VM, it brings the user's own workspace image from Z!BootOS(④) and complete the booting. After the booting-up, additional Windows application programs are streamed. (⑦).

F. The student will keep working on the desktop and logout. Then all the works on the desktop will be updated to Z!Sync and stored in Z!Storage server accordingly.

4. Experiments

4.1. Experiment Environment

The experiment is done on an environment of host servers in the computer center and a PC lab with 30 PCs in Korea Aerospace University. They include 100Mbps server-client's PC network, and 10Gbps backbone network. Windows XP was the default OS, and Z!Stream, Z!BootOS, Z!Desktop, Xenclient, etc. were installed for the experiment. Hardware specifications are as shown in Table 1. Note that VM resource allocation specifications running on the hypervisored PC are 1core, 1.36GB MM, 20GB each.

Table 1. System Hardware Specifications

	Z!Desktop Server	Z!BootOS	Z!Synch	30 hypervisored PCs in the lab (PC1, PC2)
CPU	Intel® Xeon® CPU E3-1230 V2 @3.30GHz	Intel Pentium D830 3Ghz	Intel Core 2 Duo	I7 2.93GHz 1 core
RAM	16 GB	3GB	2 GB	4GB
HDD	3.6 TB SATA	160GB SATA2	1.5 TB SATA	300 GB SATA
NIC	1Gbps NIC			

One of the objectives of the experiment is quantitatively measuring the extensibility by workspace migration and how much scale-up the VDIs on client PCs. The measured data include the time required for migration, for remote network booting, for provisioning and for storing workspace images. In addition, it is wondered that how many VDIs can be run concurrently on a single hypervisored PC in the physical lab.

Table 2 shows the elapsed time of preparing a workspace for migration in A, B, and C in Section 3.3. When the service is ready, the elapsed times for bringing a workspace from Z!BootOS server to local hypervisored PC and booting the OS are measured. It took 115

seconds until the login screen popped up. After login, it took another 47 seconds to run visual studio 2008 by streaming from Z!Stream. So, the total elapsed time to start a seamless programming work on a hypervisored PC is 156 seconds (no.4 in Table 2) which is quite acceptable compared with the elapsed time of 43 seconds to run the same application on a stand-alone PC (no.7 in Table 2). When 2 VDIs running concurrently on hypervisored PC2 and connecting to each of the VDI from RDP on a smart phone each, the elapsed time to login WinXP is 98 seconds and launch time of VS2008 is 84 seconds.

Table 2. Measured data of workspace migration time

(time units: seconds)

#	Functions	Elapsed Time	Windows XP login	VS2008 launch
1	Image upload time from PC1 to Z!Sync Server (line1)	874		
2	Image upload time from Z!Sync Server to Z!BootOS Server (line2)	112		
3	Image upload time from Z!Sync Server to Z!Desktop Server (line3)	93		
4	Elapsed time for remote booting PC2 from Z!BootOS Server (line4) and VS2008 streaming (line7)		90	66
5	RDP service connection time from PC2 to Z!Desktop Server (line5)		5.1	47
6	RDP service connection time from SmartPhone to Z!Desktop Server (line6, line8)		3.1	48
7	Elapsed time of booting OS on a Baremetal PC and application launch as a standard reference.		40	3
8	Elapsed time of running 2 VDIs concurrently on hypervisored PC.		98	84

4.2. Data Analysis

Compared the elapsed time in Table 2 with the previous NCSU VCL, theoretically it is the same if we use the same VDI solutions and hosts. Note that NCSU VCL used VMware and we used Z!Desktop which is XEN open source based. Also, compare this elapsed time with what it would take if you build-up all your compiling work environment on a new PC from scratch. The migration of workspace is faster in order of magnitude. However, if you compare elapsed time of 40 seconds to boot WinXP and 3 seconds to launch Microsoft VisualStudio 2008 on a physical PC as a reference with that of 93 seconds to boot it on a virtual PC and 5 seconds to connect and another 47 seconds for streaming the application, the new model is 71% slower.

Regarding the feasibility of the hypervisored PC, it is worth comparing the elapsed times shown in 4 and 8 in Table 2. It takes 90 seconds to boot a hypervisored PC with single VM and another 66 seconds to run the application. If two VMs on the same hypervisored PC booted sequentially and run the application concurrently, it takes 98 seconds and 84 seconds each. This means is that the hypervisored PC is capable to run 2 VMs concurrently and students may not feel much difference for sharing the hypervisored PC as a vehicle of new VDI machine. Therefore, the scalability of the whole VDI system is proportional to the number of hypervisored PCs included in the VCL. Considering the virtual laboratory environment of using smart devices to connect to the VDIs in the physical laboratory with students' own workspace, the range of extra time delay is within acceptable scope.

Before introducing the new model proposed in this paper, students cannot have the same workspace which consists of OS, apps, contents on their home PCs, computer lab PCs, and host servers as a uniform way making them unable to perform equally. However, the

technologies and models mentioned in this paper enable students to migrate a whole workspace as one single image to perform an identical work environment on different PCs at school or at home. This is such a big step from the conventional ways of moving individual applications or programming contents to different PCs. Thus, the benefit is that a student's workplace can be moved around, and even if the computer system or hardware is different than what you worked on before, the work environment can be coherently connected once you boot using the same image on virtual layers. This is an unprecedented ideally seamless development environment.

5. Conclusion

The previous VCL model is quite popular to extend conventional PC labs into virtual PC labs. However, it still relies on high-cost host machines, storages, and enterprise VDIs. So, many people know that it is a good model but not affordable in practice. Therefore, a new model is required to save TCO (Total cost of ownership) by utilizing XEN, KVM open source codes, local PC hardware resources as vehicles to run VDIs, and students' smart devices as VDI terminals.

The key issues addressed in this paper are (1) how to extend the previous model to a more affordable model by scale-up VDIs on conventional PCs at physical labs and (2) whether the functionalities and performance of the new system are acceptable. From the perspective of functionality modeling, we define a workspace and propose how to create, manage, execute, migrate, and store the workspace for individuals so that students can keep and work seamlessly with all their contents, apps, and even desktops to enrich their profiles. Furthermore the model presents how to utilize PCs as a vehicle to run VDIs. The performance and scalability of virtual machines are measured; they showed that the performance of the new VCL model is the same as previous VCL if they both use the same VDI solutions and it is 71% slower compared to physical client PC. Scalability is improved linearly to the number of hypervisored PCs included. So, the model proposed in this paper is quite acceptable as a realistic model to apply to universities today in terms of affordability and performance. As a further study, this model will be applied to campus-wide early next year and then more realistic performance data will be released in near future.

References

[1] R. A. Baratto, "MobiDesk: mobile virtual desktop computing", Proceedings of the 10th annual international conference on Mobile computing and networking, ACM, **(2004)**.
[2] S. D. Burd, A. F. Seazzu and C. Conway, "Virtual Computing Laboratories: A Case Study with Comparisons to Physical Computing Laboratories", Journal of Information Technology Education: Innovations in Practice vol. 8, **(2009)**, pp. IIP-55-78.
[3] S. D. Burd, G. Gaillard, E. Rooney and A. F. Seazzu, "Virtual Computing Laboratories Using VMware Lab Manager", Proceedings of the 44th Hawaii International Conference on System Sciences, **(2011)**.
[4] K. S. Kwan, "Direction and Issues of Smart Education in Universities", Ministry of Education, Korea, **(2013)**.
[5] S. Ouf, "An enhanced e-learning ecosystem based on an integration between cloud computing and Web2.0", Signal Processing and Information Technology (ISSPIT), IEEE International Symposium, **(2010)**.
[6] H. E. Schaffer, S. F. Averitt, M. I. Hoit, A. Peeler, E. D. Sills and M. A. Vouk, "NCSU's Virtual Computing Lab: A Cloud Computing Solution", IEEE computer, vol. 42, no. 7, **(2009)**.
[7] D. Song, Y. Kim and Y. In, "Virtual Computing Laboratories Extension with Virtual Desktop Infrastructure for Smart Campus on a Cloud Computing", Advanced Science and Technology Letters, vol. 43, (Multimedia 2013), http://dx.doi.org/10.14257/astl.2013.43.10, **(2013)**.
[8] V. Soundararajan, "Challenges in building scalable virtualized datacenter management", ACM SIGOPS Operating Systems Review archive, vol. 44, Issue 4, **(2010)**.
[9] W. Tian, "A Framework for Implementing and Managing Platform as a Service in a Virtual Cloud Computing Lab", International Workshop on Education Technology and Computer Science, **(2010)**.

[10] M. Vouk, S. Averitt, M. Bugaev, A. Kurth, A. Peeler, H. Shaffer, E. Sills, S. Stein and J. Thompson, "Powered by VCL" - Using Virtual Computing Laboratory (VCL) Technology to Power Cloud Computing", Preliminary Proceedings of the 2nd International Conference on Virtual Computing Initiative, RTP, NC, USA, **(2008)**.

Authors

Dongho Song

He is a Professor at the Department of Computer Engineering, Korea Aerospace Univ. Seoul, Korea. His research interest is in the areas including cloud computing and distributed operating systems. He holds a Ph.D. in Computer Science from Univ. of Newcastle in England. He founded SoftonNet Inc. in 1999 and has been working on virtualization technologies. Over 10 years before founding the company, he had worked on development and application of cutting edge computing and information technologies in Stanford Research Institute (SRI) in USA as well as ETRI (Electronics and Telecommunications Research Institute) in Korea.

Ji-Ae Shin

She received a Ph.D. in Computer Science from New York University in USA. Her research interests include Automated Planning and AI-related application areas such as Semantic Technologies, Data Mining and Information Integration.

In, YeanJin

He is a co-founder of SoftonNet and one of creators of Z!Cloud technology. As a software architect with more than 10 year experience, he leads research and development in SoftonNet. His specialty includes distributed systems and cloud computing. He holds a master degree in Computer Science from Korea Aerospace University.