pVEE: a Personalized Virtualized Experimentation Environment for Education Based on Virtual Machines

Abstract. With the development of teaching method in higher education, traditional computer labs cannot meet the needs and expectations of modern curriculum. This paper proposes a personalized virtualized experimentation environment (pVEE) which aims to provide VMs for the students as curriculum-oriented specialized experimentation environments. We layer virtualization technology (KVM/Xen) on hosts in the computer lab and implement management tools to construct the pVEE system. pVEE allows a teacher to customize a course related base image which contains dedicated operating systems and applications. A student's personalized VM boots from the course specific base image combined with a private virtual disk image including his personalized applications and user data. pVEE supports VM accesses and uses for users inside and outside of the lab. We implement interfaces for users to access the pVEE services. pVEE has been deployed in Peking University to support the course projects of a system virtualization courses. Finally, surveys of pVEE users show that pVEE can meet students' demands for personalized experimentation environments with high resource utilization ratio and manageability.

Keywords. Personalized VM, pVEE, virtualization, educational, management

1 Introduction

Campus computer lab is an important place where teaching activities and experiments are carried out. It needs to provide high quality services for involved participants (students, faculty members and administrator), providing students with convenient methods to use his experimentation environment, offering efficient ways for faculty members to manage teaching activities and making the whole system under the control of the administrator. However, with the development of teaching method, the shortcomings of traditional computer lab are becoming more and more obvious. Firstly, traditional computer lab always faces with security threats [2, 9, 14]. Users' private data and files can't be protected effectively on a public shared-computer. Because installation of extra software applications is usually not permitted in a public lab, students' enthusiasm declines and the administrator's workload increases. Secondly, the resources of computer systems could be utilized more effectively if computing resources can be provided for users outside of the lab. Lastly, few mechanisms have been provided for faculty members to participate in the management of computer lab, which makes teaching activities inconvenient to be carried out. We consider that placing the lab under the supervision of both faculty members and the administrator will gain more benefits.

A traditional computer lab cannot properly meet the diverse needs and expectations from students and faculty members, so we propose deploying virtualization techniques in computer labs, to solve the problem. Using a virtual machine (VM) as experimentation environment gains more benefits than using a physical machine for several reasons. A virtual machine (VM) is a fully protected and isolated system running above the underlying physical hardware. Users could execute, develop, and test applications without ever having to fear causing a crash to the systems used by other users on the same computer [19]. Users can rely on their personally-owned VMs as experimentation environments and freely install personalized software and store private data in the VMs. As each VM has its own virtual disk image, user data could be preserved safely. When facing unintended hardware problems, VM could be migrated easily to other physical machine to guarantee the availability of services. As multicore architecture becomes popular, hosting multiple VMs on one physical machine utilizes resources efficiently, thus the lab can supply services for more users with limited hardware resources.

There are numerous virtualization platforms like KVM [5, 6, 15], Xen, VMware, Bochs, Hyper-V. Even though virtualization techniques like VMware has been introduced in experimentation environment for educational purposes, these techniques are not curriculum-oriented and they lack of dedicated interfaces for faculty and administrator to maintain course related experimentation environments. We present an experimentation environment based on KVM/Xen, named *personalized Virtualized Experimentation Environment* (pVEE) which provides personalized VMs for students. Additionally, we design and implement convenient and efficient tools for management of the whole system.

The rest of this paper is organized as follows: Section 2 introduces the architecture design of pVEE; Section 3 describes the detailed implementation of pVEE; Section 4 introduces the deployment of pVEE in Peking University; Section 5 discusses the related work about virtualized experimentation environment from theoretical and practical perspectives; Section 6 concludes the paper and presents our future work.

2 Design

2.1 Design goals

To face the challenges that traditional computer lab brings, we come up the idea to build a virtualized experimentation system called pVEE. The design goals are as follows.

Personalized: students are allowed to do anything in the VMs provided by pVEE. They can install software and save the private data just like using their personally owned laptop. These changes will be recorded to their personalized image. pVEE provides convenient ways to access the VMs through a client or a browser.

Curriculum oriented: teachers are allowed to customize dedicated experimentation environment for their courses. pVEE allows them to choose preferred OS and applications to generate course related virtual disk images. The course-enrolled students can get the curriculum-oriented experimentation environment. Curriculum-oriented design gains many benefits. For example, teaching Computer Security labs in traditional ways is difficult and expensive because it needs a dedicated testbed network but pVEE can solve the problem by providing virtual networks.

Portable: students can use removable device (e.g. USB flash drive) to preserve their personalized VMs and resume the working environment containing operating system and applications anywhere.

Remote access: pVEE provides students outside of the lab with convenient ways to use VMs remotely. pVEE provides unified mechanisms to operate VMs.

Monitoring and management: pVEE provides the administrator with dashboards to view the run-time status of the whole system to simply forecasting, detecting and troubleshooting performance and capacity bottlenecks.

2.2 pVEE Architecture

To achieve the design goals, we propose the pVEE architecture as shown in Fig. 1. When students come in the computer lab, conducting course experiments, they can choose any virtualized host (based on KVM/Xen) to run their own personalized VMs. The VM boots from two images: one is the base image which is course related; the other one is the personalized image. The base image contains an operating system and a few necessary programs required by the course. The personalized image contains the student's private user data and personalized applications. The course related base image is stored in host's local image repository. The personalized image is stored in a centralized storage server and organized as an image pool. When students are outside of the lab, they can remotely access their personalized VMs running on hosts. pVEE provides efficient and convenient VM access mechanisms for users. We design a pVEE management tool, which contains three modules: the image module, the VM module and the metadata module. We also design interfaces for students, teachers and administrator to access pVEE services supported by the modules. The pVEE management tool will be deployed in the pVEE server.

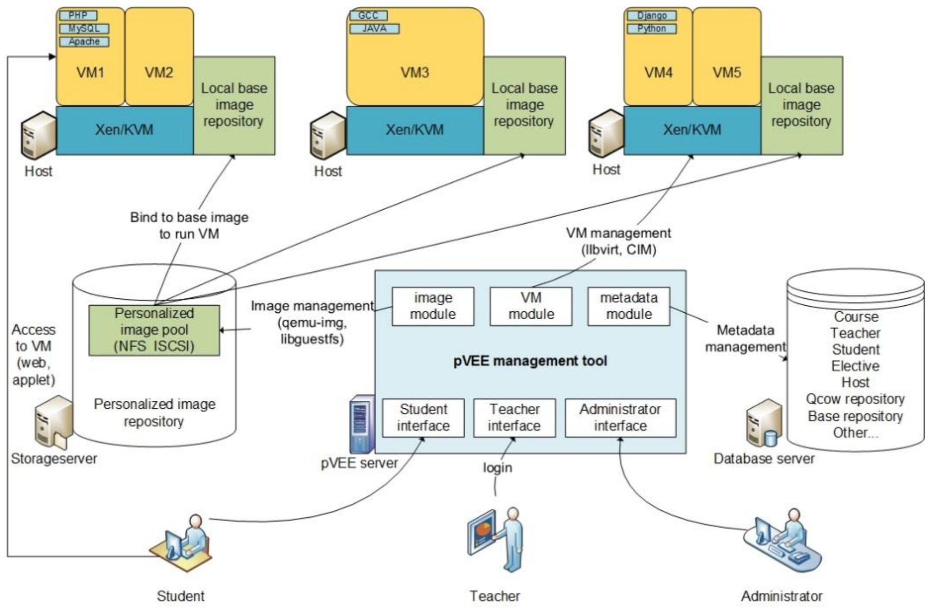


Fig. 1. Overview of the pVEE architecture design

The image module implements mechanisms to manage local image repositories in hosts and personalized image repository (pool) in the storage server. It also maintains file system inside the images.

The VM module helps to manage all the VMs running on hosts. It implements the mechanisms to switch VM running status, create VM snapshots and so on. VM module provides in-house and remote users with a unified mechanism to manage VMs.

The metadata module is designed to maintain the relationship among all the objects in pVEE. The objects (e.g. students, teachers, hosts, VMs) and relationship (e.g. a student participates in one course is an elective relationship) among objects are stored in the database server.

The modules cooperate with each other to optimize pVEE manageability. The image module provides a base image and a personalized image for the VM module to boot a VM. The relationship among the student's personalized VM managed by the VM module, the course-related base image and the personalized image of a student managed by the image module, is maintained by the metadata module.

According to various requirements from students, teachers, administrator, the pVEE management tool also provides three dedicated interfaces for the tree kinds of users: Students can participate in some courses, use the personalized VMs to carry out curriculum oriented experimentations and manage their own VMs; Teachers can manage their courses and initialize curriculum oriented experimentation environments for the students; And the administrator can monitor and manage all the objects in the pVEE system through the graphical user interface provided by pVEE management tool.

3 Implementation

To establish pVEE, in addition to the hardware in a conventional computer lab, such as hosts, storage servers, hubs, switches and cables to connect all the equipment, our main focus is to implement the modules to manage the objects in pVEE, and the interfaces for participants to access the services. In this section, we describe the ideas and techniques behind the system, through introducing how students, teachers and administrator are involved in pVEE.

3.1 Teacher's course image management

Teachers mainly participate in the management of course and course related base images. The base disk image that contains course required operating system and applications, is called course related base image template (CBIT). To create the CBIT, there is a series of steps: first, log in the pVEE server; second, manipulate basic course related information (e.g. register one course, update the course introduction and add student users to enroll in the course); and then finally prepare CBIT. The CBIT preparation means CBIT creation and CBIT distribution.

In order to efficiently create CBIT, we introduce the concept of base image template tree (BIT tree). The design of BIT tree also achieves space-saving and easy management of images. As shown in Fig. 2, the tree is defined as a collection of nodes. Each node contains a certain operating system and zero or more applications except the root node. The root node is an empty node. A depth 1 node only contains the commonly released operating system distribution by OS vendors. They may differ from each other in version. To generate each depth k $(k \geq 2)$ node, we clone its depth $k - 1$ parent node and install additional applications.

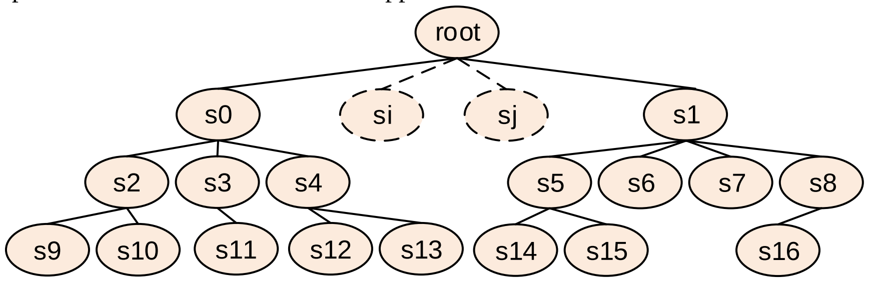


Fig. 2. Example of the base image template (BIT) tree

The teacher prepares the course base image template on the basis of the BIT tree. Firstly, the teacher submits a query, consisting of the requirements for OS and applications, to the pVEE server. Then the image module will search the BIT tree. This process returns a "yes" if there exists a 100 percent matching template, so there is no need to create a new course base image. Otherwise, it recommends a template which can best match the course requirements to the teacher for further configurations (e.g. install some more applications). The BIT tree only has depth 1 nodes initially. By studying on the previous courses and experiments, administrator creates some base image templates and updates the tree. If the query fails, the teacher has two other ways to prepare the course base image template as follows:

The teacher can prepare CBIT on local laptop by himself from scratch, and then upload the template to the pVEE server; the image module can handle numerous types of image format like RAW, VDI, etc. The image module can convert from an uncommon image format to a supported format. pVEE then updates the BIT tree by tree searching and matching, inserting this BIT node to the tree if the teacher uploads a new image template that never exists in the BIT tree before.

The teacher can also configure CBIT online based on the recommended BIT. He just boots a VM based on this image template, and then install some additional application(s) to accomplish the CBIT preparation. Finally the BIT tree gets updated by adding a child node to the recommended node.

When the CBIT preparation completes, the template will be distributed to hosts to store in their local image repository. The image module maintains the templates like this: */template/teacherID/courseID/templateName.img*. The metadata module updates the relationship among the course base image template, the course and the teacher.

3.2 Student's personalized VM

Students are mainly concerned about the user experience of their personalized VMs in pVEE. As booting VM requires CBIT and a student's course related personalized image (CPI), when the student boots his VMs for the first time, pVEE image module creates a Qcow2 [7] image as CPI for him to initialize a personalized experimentation environment. As shown in Fig. 3, there are two ways for students to use VM services: via the client and via a browser.

There are four steps when using client to access a VM: first, log in the pVEE server; second, select a course and list the course related personalized VMs; third, switch VM running status (e.g. power on VM); fourth, connect to the VM desktop via the Spice client or VNCviewer.

And there are five steps when accessing to a VM via a browser: first, log into the main page of the pVEE web service and connect to the pVEE server; second, select a course and list VMs; third, switch VM running status; fourth, the pVEE fetches the VM desktop from the host where the VM is running to the pVEE server by using cross-domain techniques; fifth, operate the VM desktop in the browser.

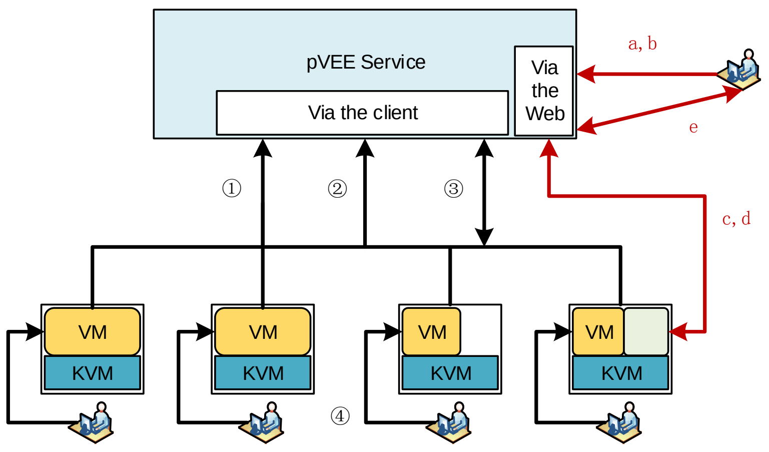


Fig. 3. Student's VM usage via the client and via the Web

There are three challenges about VM usage: how to maintain personalized Qcow2 images, how to manage VMs and how to connect to a VM desktop.

Maintenance of personalized Qcow2 images has two meanings: one is maintenance of relationship among Qcow2 images, students and courses, and the other one is management of the Qcow2 image file system. We implement the image module to achieve the goals. When a student initializes the course experimentation environment for the first time, the image module will create a CPI based on the CBIT stored in the local base image repository and store it on the storage server under */qcow2/studentID/courseID/qcow2Name.qcow2*. Then the Qcow2 image pool in the storage server exports this CPI to the user to boot a VM by calling libvirt [5] APIs. The image module sends messages to the metadata module to update the records related to the CPI. The image module integrates libguestfs, a set of APIs to access and modify the VM image file system, and integrates some useful third-party tools such as qemu-img who can automatically convert disk image formats.

When talking about VM management, we mean monitoring and preserving VM status, management of VM snapshots, and VM migration, etc. We implement the VM module with libvirt APIs integrated to solve these problems. In addition, the VM module can support various virtualization platforms like KVM, Xen and so on.

There are a variety of methods to connect to a VM desktop. When students use clients to access to their personalized VMs, they can use tools like SSH, Spice client, VNCviewer and X-Windows to connect to VM desktops, and we pre-install Spice client on the pVEE hosts. They can also use a pVEE client which has integrated LibVNCClient APIs to access their VMs. As for browser users, we integrate noVNC, Spice-Html5 tools to facilitate students to access VM desktops via the Web. We implement and run a daemon on the server side to respond to user requests.

3.3 Administrator's pVEE system management

The system administrator mainly takes responsibilities for managing and maintaining the whole pVEE system. The three most important jobs are: management of images, maintaining the relationship among all the objects in pVEE, and monitoring pVEE system (e.g. viewing live performance and resource utilization statistics).

Firstly, the images have two sources: the local base image repositories in the hosts and the Qcow2 image repository in the storage server. We implement the image module which helps create and maintain the BIT tree and manage (distribute, clone, etc.) the base image templates. By logging in the administrator's user interface, the administrator can also help teachers configure and deploy experimentation environment as a course related base image to satisfy some course's demands when the teacher cannot create course related images by himself, decreasing the workload of teachers. He can also manage (list, backup, etc.) the Qcow2 images with the image module.

Secondly, the administrator maintains the relationship among all the objects in pVEE. The metadata module helps the administrator manage the lifecycle of objects and the relationship among the objects when the administrator logs in the administrator interface. Fig. 4 illustrates the class diagram for pVEE. We can see that when a course is established by a teacher, a base image template is required. The administrator will verify that the existence of CBIT, and then synchronize CBIT to the local image repository on each host.

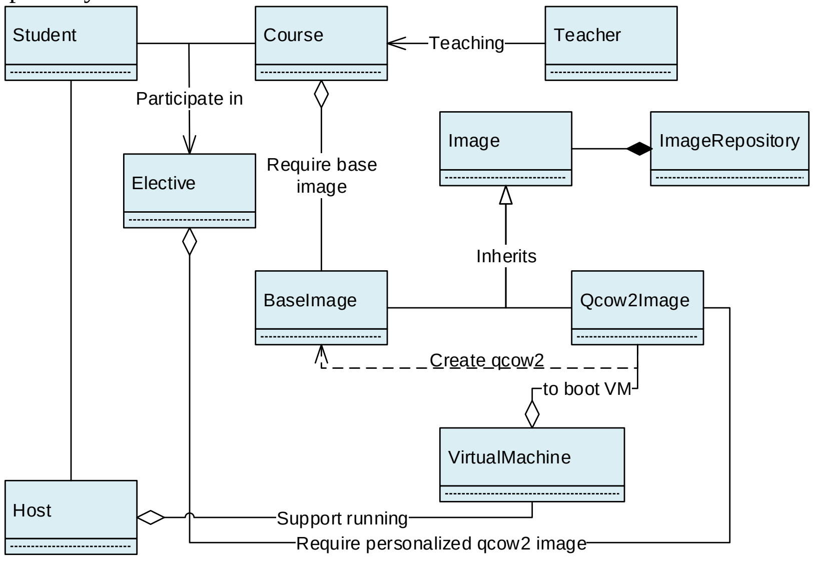


Fig. 4. Overview of the class diagram of pVEE

Thirdly, the administrator monitors the pVEE system. The VM module and the image module help report necessary performance statistics (e.g. the overhead of hosts, VM running status and disk free/usage statistics) to the administrator to gain insight in our pVEE; troubleshoot; and resolve problems before end users are affected. For example, the interface graphically displays the real time overhead (e.g. CPU, Memory consumption) of each host. The administrator can issue commands to migrate VMs to keep load-balancing when needed.

4 Deployment

We have constructed a virtualized computer lab and implemented the pVEE management tools. This section will describe the deployment of pVEE in a computer lab of Peking University.

At present, there are 22 nodes involved in the construction of the pVEE system. 20 of them are virtualized hosts. One acts as the pVEE Server and the other one the storage server to store the personalized qcow2 image files. The standalone hosts are running Centos and KVM. Table 1 shows the hardware configuration of the pVEE nodes. The software configuration of the pVEE server is shown in Table 2.

Table 1. Host hardware device capabilities and model number

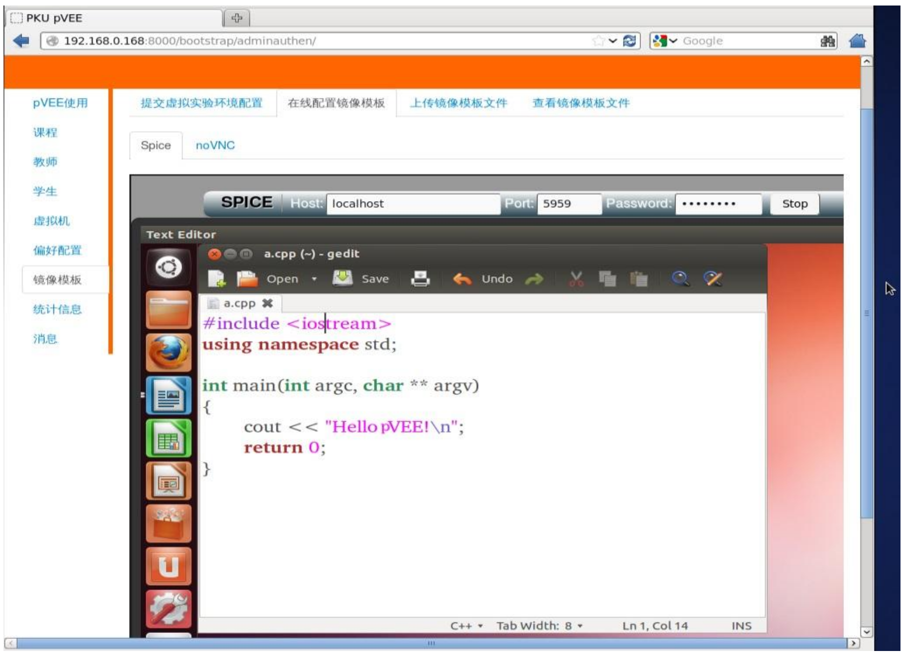
Hardware device	Model number and version
CPU	Intel i7 3700K
Memory	DDR3 32GB
Local disk	SanDisk 256G SSD + SEGATE 2T*5
Network Interface	1Gb/s

Table 2. Software configuration for pVEE

Virtualization platform	OS	Centos 6.4 X86_64
	Hypervisor	KVM 1.2.0
pVEE server	Software	Java 1.6.0 (pVEE client service) Python 2.6+Django (pVEE web service) MYSQL 5.5

At present, we carry out experiments for a course called “Storage Network and System Virtualization” using pVEE. 20 students are involved in the course. They can not only use the course related operating systems (CentOS 6.4, Ubuntu 13.04 and Window 7) and applications (e.g. Hadoop), but also install their personalized applications (e.g. QQ) and preserve private data. We get positive feedbacks from the users. The teacher considers pVEE helpful to carry out teaching activities and the students enjoy using personalized VMs to accomplish their course experiments.

Fig. 5 shows that students’ personalized VMs could be accessed via the Web. Fig. 6 shows that a VM desktop could be connected via the client.



SPICE Host: localhost Port: 5959 Password: •••••••• Stop

```
#include <iostream>
using namespace std;

int main(int argc, char ** argv)
{
      cout << "Hello pVEE!\n";
      return 0;
}
```

Fig. 5. Access to VM via the Web

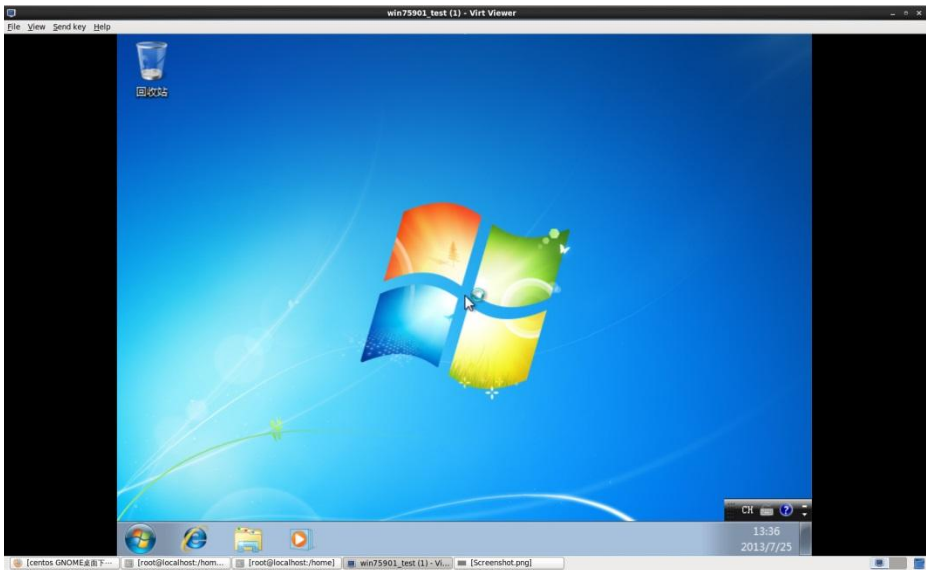


Fig. 6. Access to VM via the client

5 Related work

Numerous cases about deploying VMware virtualized desktop infrastructure (VDI) in teaching environments, aiming to increase flexibility on modern campus, are introduced in [16, 18]. The study concludes that VMWare VDI can provide very efficient and secure desktop services for students and staffs in various scenarios. Mac users can have a way to use Windows-only software, when Virtual Sites are deployed in an ITS-managed environment. VDI can also be applied to support instructional service. College of literature and arts can use older laptops to create a mobile set of lab machines for classrooms. The ILR School of Cornell University adopts Xen-based Citrix desktop to deliver VMs from a VM pool to students and staffs [17]. Some institutes consider OpenStack as an alternative to deliver VM services. Shanghai Jiaotong University adopts OpenStack to support teaching activities in its network and information center [10, 13].

The Xen Worlds project [3, 4] based on Xen virtualization creates a virtual lab environment to provide students with a personalized VM network. It promises that the off-campus students can share the VM services in an efficient and convenient way. The current Xen Worlds "server farm" consists of 5 Dell PowerEdge servers with dual-process, dual-core Xeon 1.6GHZ processors and16 or 32GB RAM, being capable of providing 470 VMs. Machine Bank [8, 11], a system engineered towards the popular shared-lab scenario, supports frontend users to preserve and resume their entire working environments. It implements a content addressable storage which benefits system performance. The ISR system developed in CMU [1, 12] provides access to a user's computing state anytime and anywhere. ISR has other features such as portable and automatic updating. pVEE is distinct from all these projects. pVEE is based on the KVM virtualization platform. pVEE mainly provides course related virtualized experimentation environments and unique management interfaces for students, teachers and administrator, which other systems do not supply.

6 Conclusion and future work

We describe how to adopt open source virtualization technology (Xen/KVM) to design and implement pVEE to solve the problems that a traditional computer lab faces. pVEE provides VMs for students; pVEE allows teachers to manage courses and provides efficient and convenient graphical interfaces for the administrator to maintain the whole system. We modularize the design of the pVEE management tools (the image module, the VM module and the metadata module). We also introduce some optimization mechanisms (e.g. base image template tree for space-saving and easy management). We have deployed the first pVEE system in a PKU computer lab. There are only 20 students enrolled in using personalized VMs now. In the future, we will optimize pVEE from the following perspectives. Firstly, we should optimize the storage and organization of Qcow2 images. Using centralized storage server to store Qcow2 images may lead to a single point of failure. When pVEE serves larger scales, the bandwidth will be a VM performance bottleneck. Furthermore, the disk space of a single file server is limited, thus we will study on the storage architecture and image organization mechanisms for optimization. Secondly, we will study more on VM guest file system. In our experiments, we found that VM performance declines when the personalized Qcow2 image size increases. So how to reorganize the file system in VM guests is another interesting and challenging work. Thirdly, we hope to support special experiments to be carried out in pVEE. For example, we will provide user interfaces for students to learn about VM kernel debugging. Fourthly, enhancing the manageability of pVEE is necessary. We will focus on optimizing pVEE from the above aspects in the future.

7 Reference

1. Gilbert B, Goode A, Satyanarayanan M. Pocket ISR: Virtual machines anywhere[R]. Tech. Rep. CMU-CS-10-112, School of Computer Science, Carnegie Mellon University, 2010.
2. Gephart N, Kuperman B A. Design of a virtual computer lab environment for hands-on information security exercises[J]. Journal of Computing Sciences in Colleges, 2010, 26(1): 32-39.
3. Anderson B R. Xen Worlds: Creating a virtual laboratory environment for use in education[J]. 2010.
4. Anderson B R, Joines A K, Daniels T E. Xen worlds: leveraging virtualization in distance education[C]//ACM SIGCSE Bulletin. ACM, 2009, 41(3): 293-297.
5. Victoria B. Creating and Controlling KVM Guests using libvirt[J]. University of Victoria, 2009.
6. Habib I. Virtualization with kvm[J]. Linux Journal, 2008, 2008(166): 8.
7. McLoughlin M. The QCOW2 image format[J]. 2008-09-11)[2010-10]. http://people. gnome. org/-markmc/qcow-image-format. html, 2008.
8. Tang S, Chen Y, Zhang Z. Machine Bank: Own your virtual personal computer[C]//Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International. IEEE, 2007: 1-10.
9. Hu J, Cordel D, Meinel C. A Virtual Laboratory for IT Security Education[C]//EMISA. 2004, 56: 60-71.
10. Zheng Q, Chen H, Wang Y, et al. Cosbench: A benchmark tool for cloud object storage services[C]//Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012: 998-999.
11. Liguori A, Van Hensbergen E. Experiences with Content Addressable Storage and Virtual Disks[C]//Workshop on I/O Virtualization. 2008.
12. Kozuch M, Satyanarayanan M. Internet suspend/resume[C]//Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on. IEEE, 2002: 40-46.
13. Zhi-cheng H. Research on Application of Open Source Cloud Computing OpenStack in Computer Rooms of Universities[J]. Computer and Modernization, 2013, 3: 053.
14. Lard K M. Creating Robust IT Security and Efficiency by Reducing Infrastructure Complexity in Higher Education[J].
15. Kivity A, Kamay Y, Laor D, et al. kvm: the Linux virtual machine monitor[C]//Proceedings of the Linux Symposium. 2007, 1: 225-230.
16. Chawla P S, Jin K, Taylor F, et al. VDI Storage overcommit and rebalancing: U.S. Patent Application 12/558,443[P]. 2009-9-11.
17. http://www.ilr.cornell.edu/techservices/services/desktopsupport/remoteaccess/citrix/

18. Ryan Henyard, Desktop virtualization in higher education. 2012
19. Chiueh S N T, Brook S. A survey on virtualization technologies[J]. RPE Report, 2005: 1-42.