4-2012

Setting Up a Low-cost Lab Management System for a Multi-purpose Computing Laboratory Using Virtualisation Technology

Heng Ngee MOK
Singapore Management University, hnmok@smu.edu.sg

Wee Kiat TAN
Singapore Management University, weekiat.tan.2009@smu.edu.sg

Citation

Mok, Heng Ngee, Lee Yeow Leong and Tan Wee Kiat. 2012. Setting up a Low-cost Lab Management System for a Multi-purpose Computing Laboratory using Virtualisation Technology. Australasian Journal of Educational Technology 28 (2): 266-278. http://www.ascilite.org.au/ajet/ajet28/mok.html

Setting up a low-cost lab management system for a multi-purpose computing laboratory using virtualisation technology

Heng Ngee Mok, Yeow Leong Lee and Wee Kiat Tan
Singapore Management University

School of Information Systems
80 Stamford Road, Singapore 178902
Email: mok@ieee.org, yllee@smu.edu.sg, weekiat.tan.2009@smu.edu.sg

Published in *Australasian Journal of Educational Technology,* 28(2), 266-278.
http://www.ascilite.org.au/ajet/ajet28/mok.html

Abstract:

This paper describes how a generic computer laboratory equipped with 52 workstations is set up for teaching IT-related courses and other general purpose usage. The authors have successfully constructed a lab management system based on decentralised, client-side software virtualisation technology using Linux and free software tools from VMware that fulfils the requirements of fast "switch over" time between consecutive lab sessions, the ability to support a wide range of IT courses and usage scenarios, low cost, easy maintenance, and a sandboxed environment for potentially disruptive IT security lab exercises. Sufficient implementation details are provided so that readers can build a similar lab management system. The objective is to share ideas and experiences that may be useful for lab administrators in academic institutions facing the same requirements and budgetary constraints.

Introduction

Hands on exercises (or lab exercises) are a significant part of IT-related courses for information systems majors at our university. Traditionally, instructors distribute the software required for running the hands on exercises to students, or direct them to websites to download the relevant installers. Students then install and run the exercises on their personal laptops. Alternatively, for exercises requiring more complex installations, instructors prepare a VMware virtual appliance containing the relevant operating system (OS) and applications for distribution. Students can then run the virtual machine (VM) using VMware Player on their laptops and complete the exercise on the VM.

Both approaches have shortcomings. Distributing installers requires students to spend time on installation, and may result in software conflicts with existing programs on their laptops. Troubleshooting corrupted installations, software conflicts, firewall problems and OS configuration settings takes time and distracts students from the objectives of the exercises. This approach is not

viable if installers require a special OS not commonly installed on students' laptops such as *Windows* server or Linux. The second approach of distributing instructor-prepared virtual appliances eliminates most of these problems, but requires students' laptops to have sufficient disk space and higher specifications, especially an abundant amount of RAM. If the software to be installed requires license keys, key distribution and subsequent removal of the software need to be policed for both approaches. Distribution of virtual appliances containing licensed application software and licensed OSs (such as *Windows*) to students needs to be very carefully considered because it is very easy for students to make copies of the *VMware* files.

In order to overcome these limitations, and to provide instructors with a third choice when conducting hands on exercises, the University decided to set up a teaching laboratory equipped with 52 modern workstations in 2008. There were five primary requirements specified for this new teaching lab:

1. Flexibility. The lab should be usable by as many courses and teaching scenarios as possible. Many educational institutions adopt a 'one lab per course' approach even if hardware requirements are identical because of the possibility of conflicting software used by different courses. A tertiary institution that one of us have taught at previously overcame this problem by installing a removable drive tray at each workstation and reserving one hard disk for each course. However, this solution suffers from wastage of disk space, and entails lots of administrative and logistical work.
2. Fast "switch over" time between lab sessions. Lessons are scheduled in 3-hour blocks with 15-minute intervals between blocks. In order for the lab to be used by consecutively scheduled classes, the workstations must be ready for the next lab session within the 15 minute interval.
3. Low cost. Both capital and operational expenditures for managing the lab need to be kept minimal. Students may be recruited to assist via ad-hoc work study contracts, but there are no plans for a full time system administrator.
4. High manageability and ease of use. Without a full time administrator, it is imperative for the lab management software to be easy to use so that instructors can be trained to operate the lab management system for their own hands on sessions.
5. Secure and controlled environment. The need for the lab to be operated under a secure and controlled environment is important for certain hands on exercises. Most exercises require students to have root privileges, but providing root access to the workstations inevitably results in inconsistent machine states, unauthorised programs being installed, potential security risks and corrupted machines. It is also preferable if the instructor can control network access from the workstations, especially for potentially disruptive hands on exercises in IT security courses involving packet sniffing and simulated attacks.

We evaluated several commercial lab management solutions including Microsoft Desktop Optimization Pack (MDOP) and various hard disk cloning tools. The cloning solutions centre round pushing down clones of entire disk drives or partitions from a central storage server during the "switch over" time between lab sessions. Unfortunately, cloning disk partitions to 52 workstations may take significant time, and none of the vendors could guarantee a 15-minute "switch over" interval. MDOP is a Microsoft desktop suite which includes tools such as Microsoft Application Virtualization (App-V) and Enterprise Desktop Virtualization (MED-V). Although both App-V and MED-V are applicable technologies, they are directed at Microsoft applications and Windows respectively. Some of the hands on exercises require Linux, which is not supported by these products. The computer lab at our university's library uses a commercial software called Deep Freeze that enables administrators to take snapshots of the workstation's configuration and settings. During a reboot, Deep Freeze restores the workstation to the previous checkpoint and therefore guarantees a consistent state for the next user (Faronics, 2011). This may apply to general purpose workstations, but not for our scenario because this system roll-back software supports only up to eight checkpoints, does not support Linux, and students would be quite frustrated to lose all their work

every time they reboot their workstations. After finding no satisfactory commercial solution, we decided to build an in-house lab management system based on free virtualisation software available from VMware.

Related work

Using virtualisation technologies for conducting hands on exercises in computer science-related courses is not new; Stockman (2003) described one of the earliest examples of using VMs to run computer networking lab exercises. Collins (2006) highlighted the advantages of using VMware for hands on exercises in security and OS courses, and suggested that lab managers and instructors consider using VMware in teaching. Stackpole et al. (2008) gave a good overview of the problems faced by using "real" OSs for lab exercises and described the advantages of using virtual lab environments to increase laboratory utilisation and student productivity. Bulbrook (2006) and Bullers (2006) summarised the advantages of using virtualisation technologies for running student hands on exercises which are relevant to our scenario: (i) ease of setup and deployment, which basically involves creating a copy of the VM files, (ii) prevention of setup conflicts between different lab sessions (each VM can be specifically prepared for a particular exercise), (iii) possibility of giving administrator rights to students without compromising security, (iv) possibility of "generic" laboratories shared by various courses rather than dedicated labs for each course, and (v) the possibility of letting students experiment with "dangerous" activities such as penetration testing, injection attacks, session hijacking and spoofing in a "sandboxed" environment without affecting the campus network.

Over the past few years, hands on exercises using VMs have been tried successfully in courses related to information, web or computational security (Tao, Chen & Lin, 2010; Gephart & Kuperman, 2010; Hay, Dodge & Nance, 2008; Hickman, 2008; Bullers, Burd & Seazzu, 2006; Adams & Laverell, 2005), operating systems (Nieh & Vaill, 2005; Adams & Laverell, 2005), *Windows* and Linux system administration (Hickman, 2008; Stackpole et al., 2008), database (Cranitch & Rees, 2009), computer forensics (Hickman, 2008; Bullers, Burd & Seazzu, 2006), data communications (Hickman, 2008) and networking (Stackpole et al., 2008; Bullers, Burd & Seazzu, 2006; Adams & Laverell, 2005; Armitage & Harrop, 2005). Most of these publications describe case studies of the use of VMs in specific IT courses and focus on the content of the exercises and how the lessons were conducted.

Li (2010b) categorised virtualisation techniques used in labs into two broad groups: centralised and decentralised. In the centralised model, the VM files are hosted on a powerful central server (or 'cloud'), and students remote in from laptops - that act as dumb terminals - into individual VMs running concurrently on the central server. In this model, the number of VMs that can run concurrently on the server (which is also the number of students who can remote in concurrently) depends very much on the resource requirements of each VM and the capacity of the server. In the decentralised model, the VM files are distributed to workstations or laptops, and students complete the exercises on the VM using a hypervisor (such as *VMware Player* or *VirtualBox*) running on the workstation or laptop. Unlike the centralised model whereby most of the work is done at the server, the decentralised model distributes the processing work to individual laptops/workstations. For the former, it is important that reasonably fast network connectivity between laptops and server be maintained throughout the lab session. For the latter, network bandwidth is required only during the distribution of VMs to the laptops. Tao, Chen & Lin (2010) provide a summary of the advantages of the decentralised model.

Documented examples of real implementations based on the centralised model include (i) the virtual computer lab project (Gephart, 2010), (ii) the virtual computing lab at North Carolina State University (Li, 2009), (iii) the virtual IT teaching lab at Bond University (Cranitch & Rees, 2009), (iv) the multi-course systems lab at Calvin's College (Adams & Laverell, 2005) and (v) the Remote Unix Lab

Environment at Swinburne University of Technology (Armitage & Harrop, 2005). Examples of decentralised implementations can be found at Pace University (Tao & Chen, 2010), Utah Valley State College (Hickman, 2008), and the virtual network lab at the University of New Mexico (Bullers, Burd & Seazzu, 2006). In these decentralised implementations, students are aware that they are running a virtual environment on their laptops or workstations. In fact, for some of the hands on exercises for OS, security and networking courses described, it is important for students to be aware of how to configure the hypervisor so that the VMs can be grouped into a virtual network.

The next section provides a description of how we adopted some of the existing virtualisation technologies, ideas and implementations to come up with a novel lab management solution based on a decentralised approach, to fulfill the requirements of our school. Justifications for certain choices we have made are explained, and sufficient implementation details are described so that readers familiar with Linux, shell scripting and *VMware* are able to build a similar system.

Implementation: Basic functionality

The 52 workstations acquired in 2008 for the new teaching lab had relatively modern specifications at that time: 2 GB RAM, 2.33 GHz Intel Core 2 Duo processors, and a 250 GB SATA hard disk drive. A centralised model will require the additional procurement of a very powerful server capable of running 52 concurrent VMs. To keep costs down, an easy decision was made to adopt the decentralised model to leverage on these relatively powerful workstations by running VMware Player locally on each of them. VMware Player for Windows and Linux is free for personal non-commercial use (VMware, 2011a). In virtualisation technology nomenclature, the term "host OS" refers to the actual OS installed on the physical hardware, and "guest OS" refers to the OS installed on the virtual hardware emulated by the hypervisor. In this paper, the term VMware"image" refers to the folder containing all the VMware files comprising the guest OS and any necessary software already installed on the guest OS (i.e. all the files comprising the virtual appliance). Depending on the guest OS and software installed, image sizes can range from below 1 GB for a bare-bones Linux image to 25 GB for a Windows Server 2008 image with SharePoint Server installed. To give a better idea of space requirements: a typical "clean" Windows XP Pro image with a little space for user data can be created with as little as 2 GB of disk space, and a "clean" Windows 7 Pro image takes up slightly more than 7 GB.

The crux of our implementation is to have one independent *VMware* image for each student and lab session. For each hands on exercise, the instructor prepares a *VMware* image which is then distributed and stored on the hard disk of every workstation. With multiple images stored on each workstation, it is possible to switch between images for the next lab session within a few minutes simply by shutting down the current image and powering up the correct one.

We decided on Linux as the host OS for the workstations for three reasons: (i) Linux is free of charge, (ii) it has very powerful scripting capabilities and (iii) in its stripped down form, Linux is relatively less resource hungry compared to any variant of *Windows*. Considering that virtualisation has a performance overhead, the last factor is significant. A stripped-down version of Ubuntu Linux with only the required OS components (including *X windows* and the necessary drivers), and *VMware Player* were installed on the workstations. For the hypervisor, we were initially considering *VMware Player* and Microsoft *Virtual PC*. The latter had to be eliminated because of our choice of the host OS. Not only does *Virtual PC* need *Windows* as the host OS, it is not free software, and does not officially support Linux as a guest OS. *VMware* was selected over other open-sourced hypervisors that run on Linux (such *Xen Hypervisor* and Oracle *VirtualBox*) because of the maturity of the *VMware* platform and our familiarity with this product. Readers can refer to Li (2010a) for a comparison between *VMware* and *VirtualBox*.

One of the workstations in the lab was designated the "server", and the remaining workstations became "clients". A data partition was created on each client workstation for storing images, and each workstation is allocated a unique static IP address which is hard coded in the */etc/network/interfaces* configuration file. At any one time, each client should only be running one of the *VMware* images from its data partition. A text configuration file called *bootup* containing information on which *VMware* image should be the "current" image is created and stored on the server workstation. *bootup* stores name/value pairs containing the IP address of each client workstation and the corresponding name of the "current" *VMware* image for that client. This makes it possible for different client workstations to have different "current" images so that instructors can run exercises involving multiple machines with unique roles (such as scenarios depicting communication between a client and a server).

Shell scripts were written to perform fundamental management tasks. These scripts are executed by the instructor at the server workstation:

- *wakeClient*: Power up specific client workstations. This script uses the Wake-on-LAN feature supported by these workstations to send "magic packets" to the list of MAC addresses of specific client workstation to turn them on if they are not already powered up.
- *shutDownClient*: Terminates all processes on specific client workstations, including *VMware Player*, and shut down the machine gracefully.
- *checkOnline*: Check if specific client workstations are alive and responding.
- *deploy*: Copy images from a shared folder on the server workstation to the data partition of specific client workstations during the image distribution process. Two different scripts were written for deployment: the first uses secure copy (SCP), and the other uses the BitTorrent protocol (using cTorrent binaries). In the SCP approach, all client workstations which have received a complete copy of the image will initiate the copying process to another client workstation that is waiting, until all client workstations have a copy of the image. In the BitTorrent approach, the server workstation is the initial seed and tracker node, and client workstations become leechers/peers. The BitTorrent approach appears to exhibit slightly better performance than SCP.
- *setClientVm*: This script modifies the *bootup* configuration file on the server workstation so that the instructor can change the "current" *VMware* image for specific client workstations.
- *rebootClientVm*: This script reboots the *VMware* image running on specific client workstations. When *VMware* starts again on each client workstation, it checks the bootup configuration file on the server workstation and executes the latest "current" image.
- *netDown*: prevents the guest OS on specific client workstations from accessing the Internet.
- *netUp*: allows the guest OS on specific client workstations to access the Internet.

vmrun, which is available from VMware, is a set of command line utilities for performing various tasks on VMs (VMware, 2009). *vmrun* commands can be invoked from shell scripts, and can be used for the following tasks: power on/off or reboot the VM, run programs within the guest OS, perform screen captures, and perform file operations such as deleting files or creating new folders. Most of the scripts mentioned above use the *vmrun* utility to control the VM that is running on the client workstation. We found no straightforward way to disable Internet access from the guest OS, so the *netDown* script disables Internet access from the host OS by deleting the default gateway from */sbin/route*. This indirectly prevents the guest OS from connecting to the Internet.

The */<home>/xinitrc* startup file on each client workstation was modified so that immediately after *X windows* boots up, it performs two checks:

1. Using *rsync*, it checks the directory listing of *VMware* images on the server workstation's shared folder, and compares it with the list of images in its local data partition. Images on the local data partition that are not found on the server are deleted. This provides a simple way

for instructors to notify client workstations that certain images should be removed from their local hard disk.

2. The client tries to read the *bootup* configuration file on the server workstation for the "current" image for that particular client to start running. If there is an error accessing *bootup*, the last "current" image will be assumed to be the correct one.

The client workstation will auto-login to a user account, start up the X server, load a light-weight windows manager called Fluxbox, and run VMware Player in full screen mode with the correct "current" image. When the client workstation powers up, the user sees Ubuntu's startup display, the VMware splash screen, followed by the login screen (if user authentication has been configured in the guest OS) or the desktop of the guest OS (if no authentication is required). If the image has been prepared with the virtual Ethernet card, USB, sound and DVD/CD drivers installed, users have direct access to the network, ports and DVD/CD drives on the workstation via the guest OS. It would seem as if the workstation has booted directly into the guest OS, and a lay user would have absolutely no idea that the workstation is actually a Linux machine.

VMware Player comes with *VMware Tools*, a suite of utilities that improves performance of the guest OS and provides several useful management functions. From experience, a *Windows VM* without *VMware Tools* installed may experience mouse stuttering and sluggishness, and we strongly recommend that *VMware Tools* be installed in the guest OS. The *VMware Tools* icon that appears on the *Windows* task bar in the guest OS can be hidden. In fact, nothing gives the user a clue that he/she is actually using a guest OS at the client workstation except when he/she checks the list of running processes or checks the list of installed programs from the *Windows Control Panel*.

Once the guest OS is powered up in full screen at the client, there are only two ways the user can access the underlying Linux host OS: (i) by activating *VMware's* escape key combination (defaulted to *ctrl+alt*) or (ii) by switching to a Linux terminal screen (by hitting *ctrl+alt+F1-6*). Because the *ctrl+alt* key combination is easily activated by accident, we changed it to a more complex sequence (It is possible to change the escape key sequence to various combinations of the *ctrl, alt, shift, up* and *down* keys). For this implementation, we removed all the icons and task bars on the *X Windows* desktop of the host OS, and disabled the context menu that usually appears when the user right-clicks on the desktop. If the user is aware of the *VMware* escape key combination and activates them, what appears on the screen is a plain black desktop with only a window showing the guest OS. On the other hand, if the user switches to a terminal screen, he/she would need to provide a local Linux user ID and password before executing any Linux command. In this way, it is virtually impossible for the user to "escape" from the guest OS and interact with the host OS directly. The inability to interact with the host OS also means that the user is unable to extract any of the images on the hard disk. If the image contains software that is licensed on a per-seat basis, this ensures that the image cannot be duplicated and distributed by the user. The disadvantage is that students cannot continue their work from home, and will need to go back to the same workstation to complete the exercise if he or she is unable to do so during the allotted lab period.

A few days before the hands on session, the instructor prepares a *VMware* image using *VMware Player*, uploads it to the shared folder on the server workstation, and deploys the image to all client workstations using the*deploy* script. During the "switch over" interval before the lab session, the instructor logs into the server workstation, uses the *setClientVm* script to set the "current" image, and runs the *wakeClients* script to boot up all the client workstations. Since changes on each image on individual client workstations are persistent, students can continue with their exercises during a subsequent session as long as the instructor configures the clients to boot up to the same image. In order to remove images from client workstations, the instructor deletes the image from the server's shared folder. During the next boot up, clients will detect this change and remove the unwanted images from their data partition.

For our implementation, virtual network computing (VNC) server software was installed on the host OS of every client workstation to give instructors the choice to monitor or control a client's screen. Instructors simply run a VNC client on the server workstation or any laptop connected to the lab's network, and provide the password in order to view the display screens of individual client workstations. Free VNC clients are available for*Windows* and Linux, and some of these VNC clients allow multiple content from more than one VNC server to be streamed over for concurrent display. This also gives instructors a simple way to display a particular client's screen on the projector for instructional purposes. Figure 1 shows the software stack running on each of the client workstations.

This solution went "live" in 2008, and informal feedback was collected over four semesters of usage from students and instructors who used the lab.

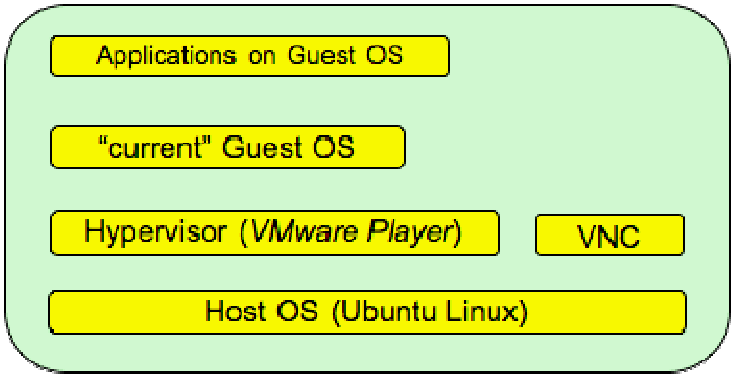

Figure 1: Schematic showing software stack running on each of the 51 client workstations

Implementation: Enhancement

The main complaints from student users are the perceived slow reaction of the workstation to user actions and occasional guest OS "freezes" when running resource hungry images. In 2010, the lab management system was revamped to rectify the performance issue and include several new features based on user comments. Hardware changes included increasing the amount of RAM on each workstation to 4 GB and replacing the 10/100 Mbps Fast Ethernet switch used in the lab with a Gigabit Ethernet (GbE) switch. The 2 GB physical RAM limits the virtual RAM on each image to approximately 1.6 GB (anything close to 2 GB would cause excessive paging which results in disk thrashing, and a significant drop in performance). The RAM upgrade allows guest OSs to be configured with near to 4 GB of virtual RAM for exercises requiring more memory, without resulting in thrashing. With the previous 10/100 switch, deploying a 5 GB image to all clients took approximately two hours. The same job could be done within 20 minutes with the new GbE switch.

Three new features were also added: (i) choice of user to choose which image to start at the client workstation, (ii) ability to run two images concurrently on a client workstation, and (iii) new scripts to insert and retrieve files from the guest OS currently running on the client workstations.

The first new feature allows a user to select one of the images already on the client workstation that he/she wants to start. We use the term "auto mode" to depict the original boot-up behaviour of the client workstations (whereby the instructor sets the "current" image that the client workstation should run automatically), and "kiosk mode" to represent the new feature. When the clients are operating in kiosk mode, users see a menu showing the list of images that can be "booted" into. The user chooses one of them and *VMware Player* runs the selected image in full screen. The administrator runs a script called *setKioskMode* on the server workstation to set the mode to auto or kiosk, as well as the list of images that the user can select from if the workstation starts in kiosk mode. The kiosk mode feature

would be useful in two scenarios: (i) when the instructor wants students to start on another exercise using a different image after completing the first one, and (ii) for usage during open hours. Open hours (or free access hours) are periods during which students can enter the lab to use the workstations without the presence of instructors. Open hours can be shared across courses, or for general purpose usage, by allowing students to select the relevant image in kiosk mode.

The second new feature introduced is the ability to run two images concurrently on the client workstation with each image appearing in its own window on the desktop. This setup is useful for IT or network security exercises that involve interactions between two machines that may affect traffic on the campus network. A particular hands on exercise in an IT security course at our school requires students to experiment with packet sniffing. Two images are concurrently loaded with one running the packet sniffer, and the other sending out packets. In order to isolate these two images from the other clients, the network adapters of both images have been set to "host only" in order to simulate an intranet within the client workstation. In order to support this feature, the *setClientVm* script has been modified to enable the instructor to select up to two images to boot when the clients are set to auto mode. In this mode, students see two windows on the otherwise blank desktop and may choose to minimise either window to an icon at the bottom of the screen, or display both concurrently. When running in this configuration, students do realise that they are actually working within a virtual environment, but they are still unable to interact with the host OS.

The third new feature uses the *vmrun* utility to manipulate files within the guest OS. Two scripts are written: *pullFromClient*, and *pushToClient*. When executed on the server, *pullFromClient* extracts the contents of a specified folder in the guest OS that is currently running, and stores them in a shared folder at the server workstation. *pushToClient* does the reverse: It copies the contents of a folder on the server workstation to a specified location in the guest OS. These scripts are useful when instructors want to retrieve assignment deliverables at the end of the lab session, or distribute files that were intentionally or accidentally omitted during image preparation.

Experience and limitations

In the past three years, this teaching lab has been used for running hands on exercises in security, business intelligence and web application courses, and practical tests for programming courses. For practical tests students are instructed to save their answers to a specific folder in the guest OS. These files are then collected to the server workstation for marking using the *pullFromClient* script. The lab has also been used as a general purpose lab when University visitors need workstations to access the Internet (a Windows 7/XP image with a browser installed converts the lab into a functional general purpose lab). Nevertheless, this lab management solution suffers from the following limitations:

1. *Concurrent usage of client workstations impossible:* Since a user's VM session is persisted on an image on individual client workstations instead of at some centralised shared storage, our implementation essentially ties a user session to a particular client workstation. It is not possible for two users to access their respective VMs concurrently to complete their half-finished exercises, even during open hours.
2. *Performance and disk space:* Virtualised platforms are unable to perform at the same levels as native platforms. We recommend workstations be fitted with as much RAM as the budget allows in order to support hands on exercises that require more virtual memory. The hard disk size on each client workstation also limits the number of images that can be concurrently deployed, so consideration should be given to the hard disk size of each workstation.
3. *Graphics capabilities, driver and technology support:* There will always be a time gap between the release of the latest OSs or new technology and the release of a version of *VMware Player* that supports them. For example, machines with built-in *Bluetooth* support have already been on the market for a few years, but *VMware Player* 4.0 that allows Bluetooth devices on the host to be shared with a *Windows* guest was only released in

October 2011 (VMware, 2011b). The range of graphical applications that can run within a guest OS is also constrained by the virtual drivers that come with *VMware Player*. Games and graphical simulators that require high performance graphics capabilities are not likely to run on the guest OS. One of the authors had planned to use the lab for a post-exam, first-person shooter gaming event, but realised that the proposed game (*Urban Terror*) could not run on the guest OS because of graphics driver incompatibilities. In a traditional lab, it is possible to slot in new hardware cards to ensure that a particular application runs; in our implementation, *VMware* draws the boundaries of what can be done.

4. *Special scenarios:* Another limitation is that *VMware Player* does not support the *Mac OS* as a guest OS, so institutions conducting Mac-related classes will still need a roomful of Mac machines (unlike *VMware*, Oracle VirtualBox does support *Mac OS X*, so the latter might be worth evaluating if Mac support is important). There is also a special category of software that is purposely built to run only on a native OS. An example is the *Exam Browser* used in our University. This is a custom-written web browser that locks the user to an online quiz web page by preventing them from switching to other web sites or applications until the quiz is complete. To prevent compromising testing integrity, *Exam Browser* terminates when it detects that it is not running on a native OS. There are also some special hands on exercises that do not make sense in a virtualised environment. An example is a load testing experiment that may give inconsistent performance results when the system under load runs within a guest OS.

Except for these special cases, we have found our implementation suitable for most teaching and usage scenarios.

Future work

Some ideas for future extensions are discussed here.

1. For our solution, images sent to client workstations are independent *VMware* "full clones". Stackpole et al (2008) suggested using *VMware's* "linked clone" feature and provides a brief explanation on how it works. If limited hard disk space on client workstations is a concern, it is possible to create linked clones for each image. Linked clones are made from snapshots of the parent VM, take up considerably less disk space but require the parent VM files to run. It is possible for multiple lab exercises that use the same guest OS with slightly different configurations to share the same parent VM. Linked clones are prepared for each of these lab exercises and distributed to every client workstation. These linked clones share a common parent VM residing on some shared centralised storage. This configuration will require faster bandwidth to facilitate communication between the linked clone and parent VM during the lab session, and very fast disk access to the parent VM. The inevitable drop in performance may not be worth the benefit of savings in disk storage space, but it could be interesting to try this out.
2. Users may be authenticated by the guest OS using common local credentials, or credentials on a service directory (such as *Active Directory*) if the guest OS has been configured to be a member of a *Windows*domain. Our implementation suffers from the inability to easily associate a domain user ID with a particular client workstation. *VMware's* virtual Ethernet adapter can be configured to "host only", "NAT" or "bridged" mode. In "NAT" mode, the guest OS shares the same IP address as the host, but in "bridged" mode, the guest OS gets its own unique IP address on the network. In "bridged" mode, with the guest OS configured to obtain an IP address dynamically using DHCP, it is difficult to associate an IP address with a particular client workstation. Therefore it may be a good idea to authenticate users at the host OS using a directory service instead. This will ensure that it is always possible to associate a user ID with a particular client workstation, regardless of which image the user runs, whether the guest OS has been configured for authentication, or how the guest OS's virtual Ethernet adapter is configured. Such information may come in useful when tracking security breaches.

3. To bypass the limitations of hands on exercises that cannot run in a virtualised environment, an idea worth considering is to create a new partition on each workstation's hard disk, install a *Windows* client OS (e.g.*Windows 7* or *Windows XP*) natively on this partition, and configure each client workstation for dual-boot. Users can then boot into this *Windows OS* for special hands on exercises or scenarios that cannot run within a virtualised environment.
4. Some instructors who may not be familiar with the Linux shell have provided feedback that the system could be more user friendly if given a graphical user interface. A web application which acts as a fa□ade for the existing shell scripts has been planned. This web interface is expected to increase usability of the system for instructors, without affecting the existing functionalities.

Conclusion

The idea of using VMs and hypervisors in hands on exercises has been around for nearly a decade, and several publications have described positive experiences using virtualisation technology for computer science-related lab exercises. We have extended this idea to set up a lab management system that meets the requirements for fast "switch over" time between lab sessions, easy maintenance, and supporting most IT courses and usage scenarios using free software. Our multi-purpose lab abstracts the virtualisation from users, and in most circumstances, lab users sitting in front of each client workstation are not aware - and do not need to be aware - of the virtualisation software stack running beneath. Most of the documented implementations focused on the content of the exercises, and the advantages of using virtualisation technology to implement them. This paper focuses on how we implemented a decentralised virtualisation solution for a wide range of usage scenarios. We hope that this work can act as a concrete reference for lab managers and course designers who face the same requirements as we did.

Acknowledgments

We would like to thank Steve Miller, Dean of the School of Information Systems at our university for his support for this project. Kar Way Tan and the anonymous reviewers provided constructive criticisms on this manuscript. Kelvin Law was the first student who set up the original implementation and contributed very good ideas that shaped our final solution. Juniper Network donated two GbE switches to the laboratory.

References

Adams, J. C. & Laverell, W. D. (2005). Configuring a multi-course lab for system-level projects. *SIGCSE'05, Feb. 23-27, Association for Computing Machinery*. http://dx.doi.org/10.1145/1047124.1047509

Armitage, G. & Harrop, W. (2005). Teaching IP networking fundamentals in resource constrained educational environments. *Australasian Journal of Educational Technology,* 21(2), 263-283.http://www.ascilite.org.au/ajet/ajet21/armitage.html

Bulbrook, H. (2006). *Using virtual machines to provide a secure teaching lab environment.* [White Paper]. Durham Technical College. Durham.http://www.infosecwriters.com/text_resources/pdf/Virtual_Machines_HBulbrook.pdf

Bullers, W. I. Jr, Burd, S. & Seazzu, A. F. (2006). Virtual machines - an idea whose time has returned: Application to network, security, and database courses. *SIGCSE Bulletin,* 38(1), 102-106. Association for Computing Machinery. http://dx.doi.org/10.1145/1124706.1121375

Collins, D. (2006). Using VMWare and live CD's to configure a secure, flexible, easy to manage computer lab environment. *Journal of Computing Sciences in Colleges,* 21(4), 273-277. Consortium for Computing Sciences in Colleges. http://dl.acm.org/citation.cfm?id=1127439

Cranitch, G. & Rees, M (2009). Virtualization: A case study in database administration laboratory work. In *Same places, different spaces. Proceedings ascilite Auckland 2009.*http://www.ascilite.org.au/conferences/auckland09/procs/cranitch.pdf

Faronics (2011). *Deep Freeze Enterprise.* http://www.faronics.com/en-uk/enterprise/deep-freeze_en-uk-2/

Gephart, N. & Kuperman, B. A. (2010). Design of a virtual computer lab environment for hands-on information security exercises. *The Journal of Computing Sciences in Colleges,* 26(1), 32-39. Consortium for Computing Sciences in Colleges. http://dl.acm.org/citation.cfm?id=1858457; also http://www.cs.oberlin.edu/~kuperman/research/papers/xenlabs2010ccsc-mw.pdf

Hay, B., Dodge, R. & Nance K. (2008). Using virtualization to create and deploy computer security lab exercises. *Proceedings of the IFIP TC 11 23rd International Information Security Conference,* 278, 621-635. IFIP International Federation for Information Processing. Boston: Springer. http://assert.uaf.edu/papers/virtualLabs_SEC08.pdf

Hickman, G. D. (2008). An overview of virtual machine (VM) technology and its implementation in I.T. student labs at Utah Valley state college. *Journal of Computing Sciences in Colleges,* 23(6). Consortium for Computing Sciences in Colleges. http://dl.acm.org/citation.cfm?id=1352419

Li, P., Toderick, L. W. & Lunsford, P. J. (2009). Experiencing virtual computing lab in information technology education. *Proceedings of the 10th ACM conference on SIG-information technology education,* 55-59. Association for Computing Machinery. http://dx.doi.org/10.1145/1631728.1631747

Li, P. (2010a). Selecting and using virtualization solutions - our experiences with VMware and VirtualBox. *Journal of Computing Sciences in Colleges,* 25(3). Consortium for Computing Sciences in Colleges.http://dl.acm.org/citation.cfm?id=1629121

Li, P. (2010b). Centralized and decentralized lab approaches based on different virtualization models. *Journal of Computing Sciences in Colleges,* 26(2). Consortium for Computing Sciences in Colleges.http://dl.acm.org/citation.cfm?id=1858625

Nieh, J. & Vaill, C. (2005). Experiences teaching operating systems using virtual platforms and Linux. *SIGCSE Bulletin,* 37(1), 520-524. Association for Computing Machinery.http://dx.doi.org/10.1145/1047124.1047508

Stackpole, B., Koppe, J., Haskell, T., Guay, L. & Pan, Y. (2008). Decentralized virtualization in systems administration education. *SIGITE'08,* 16-18 October, 249-253. Association for Computing Machinery.http://dx.doi.org/10.1145/1414558.1414619

Stockman, M. (2003). Creating remotely accessible "virtual networks" on a single PC to teach computer networking and operating systems. *Proceedings of the 4th Conference on information Technology Curriculum,*Lafayette, Indiana. http://dx.doi.org/10.1145/947121.947137

Tao, L. & Chen, L.-C. (2010). Improving web security education with virtual labs and shared course modules. 7th Annual Research Day, Seidenberg School of Computer Science and Information Systems, Pace University. http://csis.pace.edu/~ctappert/srd2010/c1.pdf

VMware (2009). *Using vmrun to control virtual machines.* http://www.vmware.com/pdf/vix180_vmrun_command.pdf

VMware (2011a). *VMware Player FAQs.* http://www.vmware.com/products/player/faqs.html

VMware (2011b). *VMware Player 4.0 Release Notes.* http://www.vmware.com/support/player40/doc/releasenotes_player40.html#New_Features