Differential Virtualization for Large-Scale System Modeling

By

Jason Koppe

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Security and Information Assurance

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

September 26, 2008

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in
Computer Security and Information Assurance

Thesis Approval Form

Student Name: Jason Koppe
_____

Thesis Title: Differential Virtualization for Large-Scale System Modeling
_____

Thesis Committee

Name	Signature	Date

Bo Yuan
_____
Chair

Bill Stackpole
_____
Committee Member

Yin Pan
_____
Committee Member

Thesis Reproduction Permission Form

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in
Computer Security and Information Assurance

Differential Virtualization for Large-Scale System Modeling

Date: _____   Signature of Author: _____

Abstract: Today's computer networks become more complex than ever with a vast number of connected host systems running a variety of different operating systems and services. Academia and industry alike realize that education in managing such complex systems is extremely important for computer professionals because, with computers, there are many levels of detailed configuration. Configuration points can occur during all facets of computer systems including system design, implementation, and maintenance stages. In order to explore various hypotheses regarding configurations, system modeling is employed – computer professionals and researchers build test environments. Modeling environments require observable systems that are easily configurable at an accelerated rate. Observation abilities increase through re-use and preservation of models. Historical modeling solutions do not efficiently utilize computing resources and require high preservation or restoration cost as the number of modeled systems increases. This research compares a workstation-oriented, virtualization modeling solution using system differences to a workstation-oriented, imaging modeling solution using full system states. The solutions are compared based on computing resource utilization and administrative cost with respect to the number of modeled systems. Our experiments have shown that upon increasing the number of models from 30 to 60, the imaging solution requires an additional 75 minutes; whereas, the difference-based virtualization solution requires an additional three (3) minutes. The imaging solution requires 151 minutes to prepare 60 models, while the difference-based, virtualization solution requires 7 minutes to prepare 60 models. Therefore, the cost for model archival and restoration in the difference-based virtualization modeling solution is lower than that in the full system imaging-based modeling solution. In addition, by using a virtualization solution, multiple systems can be modeled on a single workstation, thus increasing workstation resource utilization. Since virtualization abstracts hardware, virtualized models are less dependent on physical hardware. Thus, by lowering hardware dependency, a virtualized model is further re-usable than a traditional system image. If an organization must perform system modeling and the organization has sufficient workstation resources, using a differential virtualization approach will decrease the time required for model preservation, increase resource utilization, and therefore provide an efficient, scalable, and modular modeling solution.

Table of Contents

Table of Figures

Table of Tables

Acknowledgements

Thanks…

To my persistent professor and advisor for life: my mother, Jennifer

To my temporary teachers and peers for life: my friends, including the RIT faculty and staff

To everyone who cares about me and my thoughts

1 Introduction

Complex computer networks exist throughout the world. In fact, computer networks are so vast and growing so quickly that colleges and universities offer degree programs focused upon computer system design and administration. An entire training industry exists for computer certifications like Microsoft Certified Professional, Cisco Career Certifications and for other major products. Further, there are even more computer certifications from organizations like SANS and CompTIA. Academia and industry alike realize that education is important for computer professionals because, with computers, there are many levels of granular configuration. Configuration points range from the way a web browser displays a page to the fashion that a network adapter queues packets for delivery or acceptance. In order to test hypotheses regarding new or different computer system configurations, system modeling occurs – computer professionals and researchers build test environments. Anti-malware researchers might setup a quarantined computer network and launch potentially malicious software to understand its behavioral traits. Software testers might setup multiple versions of different operating systems at many different configuration granularities to verify whether the software executes as expected in distinct environments. System administrators might setup a duplicate server environment to assess the latest software patches and any negative impacts they cause. System imaging and virtualization have both helped advance computer system modeling procedures and capabilities.

This research aims to determine infrastructures that employ existing resources to use and archive large-scale heterogeneous system models by utilizing workstations to perform differential operating system virtualization. By staging workstations with virtual machine templates, users can create, store, and restore differential virtual machines based on the templates. Once a user instantiates this difference, they can execute the difference using primarily the workstations computing resources.

1.1 Problem

Large-scale modeling environments require systems that are easily configurable at an accelerated rate. The cost to acquire, setup, and maintain modeling environments increases in terms of hardware and person-hours as the number of modeled entities increases. Further, historical modeling solutions do not use available resources to their upmost potential.

Models that use hardware specific imaging can lead to a model that underutilizes computing resources since each workstation executes operations for a single operating system. Unless the test requires each workstation to perform at one-hundred percent resource utilization, it is likely that a virtualization-based model could achieve higher hardware resource utilization. To elaborate, if a test requires simulated user activity similar to web browsing or document authoring, a modeled entity should not be at full utilization at the processor, network, disk, or memory. Further, these solutions, while useful in small environments and necessary for situations like hardware configuration testing and disk forensics, do not scale as the number of modeled systems increases because they require sufficient modeling hardware. For example, if a tester needed to simulate fifty workstations running Red Hat Linux and fifty workstations running Microsoft Windows 2000, a hardware specific imaging modeling approach requires the tester to have one-hundred physical workstations dedicated to the test environment – this is not practical.

Virtualization is one approach to utilize more efficiently the hardware available and thus decrease the need for excessive hardware. System virtualization, in particular, has been gaining attraction in data centers. This research doesn't aim to focus on the benefits of data-center oriented virtualization, but rather points out that centralized virtualization infrastructures outright ignore the computing resources from a pool of workstations. Many of these products do not account for the large number of high-powered workstations that are at the desk of developers, testers, researchers, students, and other computer professionals. Imagine a software firm that owns a workstation for each of its 200 engineers. Each workstation uses a 4 GHz processor and 4 GB of memory. Next, assume that each workstation cost $1,500 to purchase and deploy – putting workstation expenditures total $300,000. Now, this firm notices that virtualization products might facilitate their testing phases and have to make a decision between workstation-based or centralized virtualization. How do you replicate the distributed 800 GHz of processing power and 800 GB of memory space available from the workstations at a central location? If the firm replicated the computing resources of their workstations in a server closet, they would pay more money to purchase and deploy servers and, further, would be downplaying the capabilities of the workstations. If they centralized their virtualization, then workstations would only execute non-resource intensive applications like browsers and editors; thus, underutilizing the massive amount of workstation computing resources.

Thus, explored in this research is a modeling solution that harnesses both workstation resources and virtualization.

1.2   Importance

Generally, modeling is important to research because it enables scientists "to apply quantitative reasoning to observations about the world, in hopes of seeing aspects that may have escape the notice of others" (Silvert 2001). Computer modeling is becoming much more important as systems become more complex. This research could apply to, and therefore benefit, any organization that performs computing research tasks ranging from software assurance to systems education.

1.3   Review of current research

Research regarding the administration and execution of modeling environments has been prevalent in academician-led research. Academicians are all wondering the same thing: how does one provide environments where users can apply and model computer systems concepts? Further, even if such environments are possible, how can one manage them in a low-cost fashion? This section presents previous attempts at workstation-based virtualization for computer system modeling that include minimal guest operating system support, minimal usage of differential techniques, minimal performance analysis, no system deployment techniques, and lastly, out-dated concerns of expensive monetary costs.

(Lei and Rawles 2003) raised practical cost and space concerns regarding space acquisition and computing resource utilization. The central purpose of their study was to survey performance and cost of three virtualization technologies that would enable a more practical lab environment. Their research included both quantitative methods involving performance benchmarking with different storage and virtualization technologies and qualitative methods regarding cost analysis. They tested installation time of six virtual operating systems in three virtualization platforms VMware Workstation, Microsoft Virtual PC, and Netraverse Win4Lin utilizing six storage technologies on three separate host operating systems; further, they monitored resource utilization on the host machines during these installations. Their experiment and analysis led to a conclusion that any virtualization technology coupled with a Microsoft Windows host operating system and a networked storage system was the most cost-performance effective environment to

enable applied system and networking administration learning. This study is unique in that it quantifiably measures performance of the system at multiple points of interest: host resource utilization, virtual OS installation time, and network utilization.

(Begnum et al. 2004) presented challenges their institutions experienced using traditional physical hardware to enable students to learn and apply system administration concepts. The purpose of their study was to provide an environment where students could manipulate systems from an administrative context. The authors described use of User-Mode Linux (UML) as a virtualization platform and My Linux Network (MLN) as a virtualization administration tool at university and industry environments. The authors concluded that their use of virtualization through User-Mode Linux enabled students more efficiently learn system administration concepts. The authors stated that they're approach to enabling students to apply system administration concepts need only function "as specified in the RFC's" – therefore, they weren't required to offer specific operating systems or applications, just something that "worked correctly." While their UML architecture enables system administration education in their institution, other organizations might require implementation of heterogeneous architectures including non-Linux operating systems.

Educators at the University of Cincinnati (Stockman, Nyland, and Weed 2005) faced mobility and manageability issues surrounding a small deployment of workstation-based virtualization to teach networking and system administration material. The purpose of their study was to present their findings regarding a centralized delivery of virtual machines to a lab environment including 18 physical workstations to assist student mobility and staff system management. Their experiment was centered on an Active Directory domain that included a network-attached storage (NAS) system and eighteen workstations. The NAS ran Windows Server 2003 with dual 866 MHz processing cores, 1.5GB of memory, 2Gbit Ethernet adapter and a SCSI RAID-5 storage array and the workstations ran Windows Server 2003 each with a 2 GHz processor, 1 GB of RAM, a 1Gbit Ethernet adapter and Microsoft Virtual PC. The test was to install an operating system to a virtual machine that resided on the NAS; there were three stages of workstation involvement: five, ten and eighteen workstations. The authors measured the time it took to install the operating system at each stage of workstation involvement and noted that there was no "noticeable" difference in installation time across the three stages. The authors concluded that

mobility could be achieved with a central storage for student systems and that managing base virtual machines at the central storage was much simpler than distributing the base virtual machines to the workstations. Regarding the test, the authors did not include quantifiable performance characteristics. In addition to similar infrastructure management considerations, this research methodology will contrast their use of centralized base virtual machines by de-centralizing base virtual machines. The Stockman et al study did not mention the use of linked clones. The use of Microsoft Virtual PC as a virtualization platform limits types of supported virtual operating system to Microsoft operating systems; other organizations might require implementation of heterogeneous infrastructures that include non-Microsoft operating systems.

Other educators (Vollrath and Jenkins 2004) sought to address their problem of limited physical lab space; their lab consisted of 30 computers but was required to support nearly 60 students. They experienced logistical issues regarding lab space availability and concerns regarding high-cost instructional sign-offs. The goal of their study was explore implications and cost of using virtualization to alleviate their space and sign-off problems. The study proposed the use of Microsoft Virtual PC as a virtualization platform and further utilizing the differentiation feature of Virtual PC for various procedural benefits. Vollrath, a student at the time, evaluated the feasibility of their lab assignments in their test virtual environment. Their conclusions were broad and included an out-of-lab grading process by saving student virtual machine differences to external media, in-class exams from equivalent virtual machines are probable and easier creation of lab assignments and hoped that their infrastructure would enable students to focus on management rather than installation of systems. Vollrath and Jenkins study ostensibly used Microsoft Virtual PC to support Linux and Microsoft operating systems; only Microsoft operating systems are *supported* guest operating systems as detailed in the Microsoft Virtual PC specifications (Microsoft Corporation 2007a). As stated previously, Microsoft Virtual PC might not be an option for organizations that require implementation of heterogeneous infrastructures that include non-Microsoft operating systems. Their use of virtual machine differences to lower resource cost is a novel approach that this research project aims to utilize.

(Gaspar, Langevin, and Armitage 2007) sought to debunk virtualization "misconceptions" and clarify that virtualization for IT education is cost-effective and appropriate. Other than detailing different virtualization technologies such as hardware emulators, full virtualization, and

paravirtualization, the core purpose of their study was to present their virtualization implementation, known as SOFTICE (Scalable, Open source, Fully Transparent and Inexpensive Clustering for Education). The authors state that in using open source applications (UML/MLN) and not relying on virtual disk delivery to students as in (Stockman, Nyland, and Weed 2005) makes their system more appealing and "accessible over the internet." (Gaspar, Langevin, and Armitage 2007) argue that investing computing resources for workstation-oriented virtualization is an "investment [that] will sit mostly idle and unused." Thus, (Gaspar, Langevin, and Armitage 2007) assume that institutions do not already have computing resource capacity to utilize workstation-powered virtualization. Finally, (Gaspar, Langevin, and Armitage 2007) do not address practical environments that implement non-Linux platforms.

(Stackpole et al. 2008) addressed the lack of evaluation for decentralized virtualization that supports scalable, heterogeneous environments for use in system administration education. They described the problems with a full operating system imaging solution. The crux of the paper is the proposed usage of linked clones for storage of student-customized virtual machines. The authors demonstrate that utilization of storage, network, and management resources would decrease significantly because of the differential nature of the student data. This paper is the basis for this thesis; this research aims to quantify the claims Stackpole et al. by measuring performance and documenting management procedures.

1.4 Document Outline

The remainder of this document is organized as follows. Chapter 2 presents concepts basic to understanding the research. In chapter 3, the research environment and experiments are described. Following in chapter 4, results are presented and analyzed. Finally, conclusions are drawn in chapter 5 and the research outlook is discussed in chapter 6.

2 Background

Traditionally, experimenting with computer systems meant one required either additional computer hardware. By having additional hardware, additional physical systems could be constructed and used in experiments. Boot loaders were developed to enable multi booting. Multi booting involves installing more than one operating system to a workstation. After installing more than one operating system and upon starting the workstation, one can select which operating system to execute; thus, one can experiment with multiple logical systems on one physical system. Imaging, a process of duplicating hard disk contents, proves useful in system modeling. Using disk imaging, one can preserve the state of a disk by copying the contents to another disk or by archiving it in a single file. With disk imaging, one can easily configure similar workstations to have the same disk contents and, therefore, the same operating system and software configuration. Disk imaging has been popularized through products from companies like (Symantec 2008) and (Acronis 2008) and open source solutions like (Clonezilla 2008). Further, to increase efficiency when copying the same disk image to many disks, these products harness the abilities of multicast IP transmissions. Multicasting enables a server to send one copy of the disk image to many workstations, rather than sending many copies to many workstations. By only requiring the server to access and send the image once, the server requirements are decreased. Therefore, multicast enables scalable imaging and is useful when imaging many similar workstations. However, in the end, spare hardware is costly and multi booting or system imaging does not fully utilize the workstation hardware.

Newer to system modeling is the concept of virtualization, or abstracting computer hardware. Different types of virtualization exist, but for the purposes of this research, it is important that the virtualization platform enable the concurrent execution of multiple operating systems on a single workstation. The virtualization platform used in this research, VMware Workstation, abstracts nearly all of the underlying hardware. "VMware Workstation virtualizes I/O devices using a novel design called the Hosted Virtual Machine Architecture […] that takes advantage of a pre-existing operating system for I/O device support" (Sugerman et al. 2001).

> In this architecture, the CPU virtualization is handled by the VMM. A guest application or operating system performing pure computation runs just like a traditional mainframe-style virtual machine system. However, whenever the guest performs an I/O operation,

the VMM will intercept it and switch to the host world rather than accessing the native hardware directly. Once in the host world, the [virtualization application] will perform the I/O on behalf of the virtual machine through appropriate system calls.

This type of virtualization is also employed by Xen; however, the platforms differ in how processor instructions are abstracted. Hardware-assisted virtualization, supported by AMD-V and Intel VT, offer full, consistent processor abstraction at a loss of memory throughput (Nakajima 2007) and require special hardware. In this research, VMware Workstation enables the execution of multiple virtual machines per workstations, so one can achieve a higher utilization of workstation resources. Therefore, by coupling virtualization and disk imaging, one can configure many workstations with multiple virtual machines and increase the utilization of modeling resources. To optimize resource utilization further, many virtualization products offer the ability to create differential virtual machines. (Stackpole et al. 2008) provide the following an explanation of linked clones, which are VMware's implementation of differential virtual machines, and offer insight as to how differential virtual machines optimize storage requirements.

The use of VMware's linked clones is critical to the efficient use of network and storage resources. VMware defines a linked clone as "a copy of a virtual machine that shares virtual disks with the parent virtual machine in an ongoing manner [...while...] changes to the disk of the linked clone do not affect the parent." While the size of a modified operating system image is the sum of the size of the operating system and the modifications, the size of a linked clone is merely the size of the modifications. When saving modifications, linked clones consume less storage space; therefore, linked clones more efficiently use disk space than full system images.

At a minimum, computer networks require network services that offer high-level features for basic network usability. To enable any sort of communication, computers must address others; a basic network service, such as a DHCP server, is useful because it dynamically assigns addresses to computers. The domain naming protocol (DNS) helps humans to address computers by mapping a character based name to a computer IP address. Further, any computing environment where user accountability or access control is required, identities must be authenticated;

therefore, users must prove they are whom they claim. After authentication, certain users might be privy to certain information but not others. Access control provides a mechanism by which administrators can specify which users have access to which information. Commonly, many users have access to the same information. Thus, information sharing is employed. Just as libraries share information, books, from one location, computers can share information from one location, servers. By coupling access control with information on a commonly accessible server such as a file server, administrators can restrict and permit certain users to access a single piece of information. These network features are basic requirements of any computer network; the proposed modeling environment is no exception.

3 Research Design

In this chapter, an environment that supports differential virtualization on shared-use workstations is presented. Then, three experiments using the environment are detailed. In the first experiment, this research compares a workstation-oriented, virtualization modeling solution using system differences to a workstation-oriented, imaging modeling solution using full system states; the solutions are compared based on computing resource utilization and administrative cost while increasing the number of modeled systems. The second experiment attempts to deduce storage requirements for a specific differential virtualization approach. The third experiment demonstrates the capabilities of an individual workstation to operate many virtual machines. In section 3.1, overall assumptions and limitations for all experiments are discussed. Section 3.2 details environment implementation. Sections 3.3, 3.4, and 3.5 detail the three separate experiments. The results and analyses are presented in chapter 4.

3.1 Assumptions and Limitations

This research does not intend to determine whether clustered server-grade virtualization or workstation-based virtualization is better suited for modeling. Therefore, it is assumed that this research applies to organizations that have a substantial number of workstations whose combined processing ability and memory space is underutilized.

It is assumed that there are benefits to preserving model state and that saving progress is important ability that, when technically feasible, enhances the user experience. Thus, a large portion of this environment is focused on preserving the state of a model.

In order to measure user experience with a particular environment, it is safe to assume that if users spend less time creating, saving, and restoring the model, they can spend more time working with the model. It is assumed that having more actual time to use and manipulate the model is beneficial to modeling.

The usability of a model is independent of the preparation and archival method. Therefore, it is also assumed that even though many virtual operating systems can execute on a physical workstation, each virtual operating system should behave and respond as well as the same operating system installed in a traditional, physical sense; this means that the model has realistic

usability. Additionally, software installed on models may behave differently since software can detect its operating environment. Thus, it is also assumed that modeled processes and systems do not purposely behave differently within a virtualized model.

A rather large assumption is that the administrators employ Microsoft Active Directory with Microsoft DNS, DHCP, and File Services. Active Directory offers seamless authentication and access control, two features that the shared modeling environment requires. Further, it is assumed that, since Active Directory is a highly pervasive technology, there is no need to describe a basic setup of an Active Directory instance. In addition, VMware Workstation is the chosen virtualization platform for this environment because it offers extensive guest operating system support (VMware, Inc. 2008a).

For the purposes of testing and demonstrating a research environment, principles like least privilege, role based policy enforcement, resource quotas, and other areas that require attention in a practical deployment of such a modeling solution are ignored. It is assumed that those implementing a true instance of the research environment will pay attention to many security-focused areas that this research ignores. Throughout the remainder of this chapter, configuration points that require more attention in a practical deployment are annotated.

In modeling this research, scope limitations are required. There are initial components, such as workstations, servers, network hardware, storage solutions, and operating systems, from which samples must become determined. While modifying the configuration of components in the sample might yield different performance results, this research used the standard hardware configuration from the sampled laboratory yet varies operating systems on the workstations. The existing components in the Systems Administration laboratory for the NSSA department at RIT are used. The laboratory contains twenty benches each with four workstations. The eighty workstations operate a 3.4 GHz processor, 3 GB of memory, 110 GB hard disk, and two one-gigabit network adapters. Each workstation has a one-gigabit network connection to a data subnet while the other adapter connects to the bench hub. A Cisco 6509 with one-gigabit capable ports provides the laboratory network switching and routing features. A server, noted as Jabba SRV in the diagram below, running Windows Server 2008 is equipped with 8 GB of memory, an Adaptec 2820SA RAID adapter, five hard SATA disks in a RAID 5 array, and an

Intel quad-gigabit network adapter offers networked file services for the laboratory using a four-gigabit bandwidth network team. Finally, the Active Directory infrastructure is comprised of a set of virtual machines running on a VMware ESX cluster of Dell PowerEdge 2850 rack mount servers, one of which is noted as Vader SRV in the diagram below. Figure 1 portrays the laboratory from a logical perspective; insignificant physical entities are not illustrated.
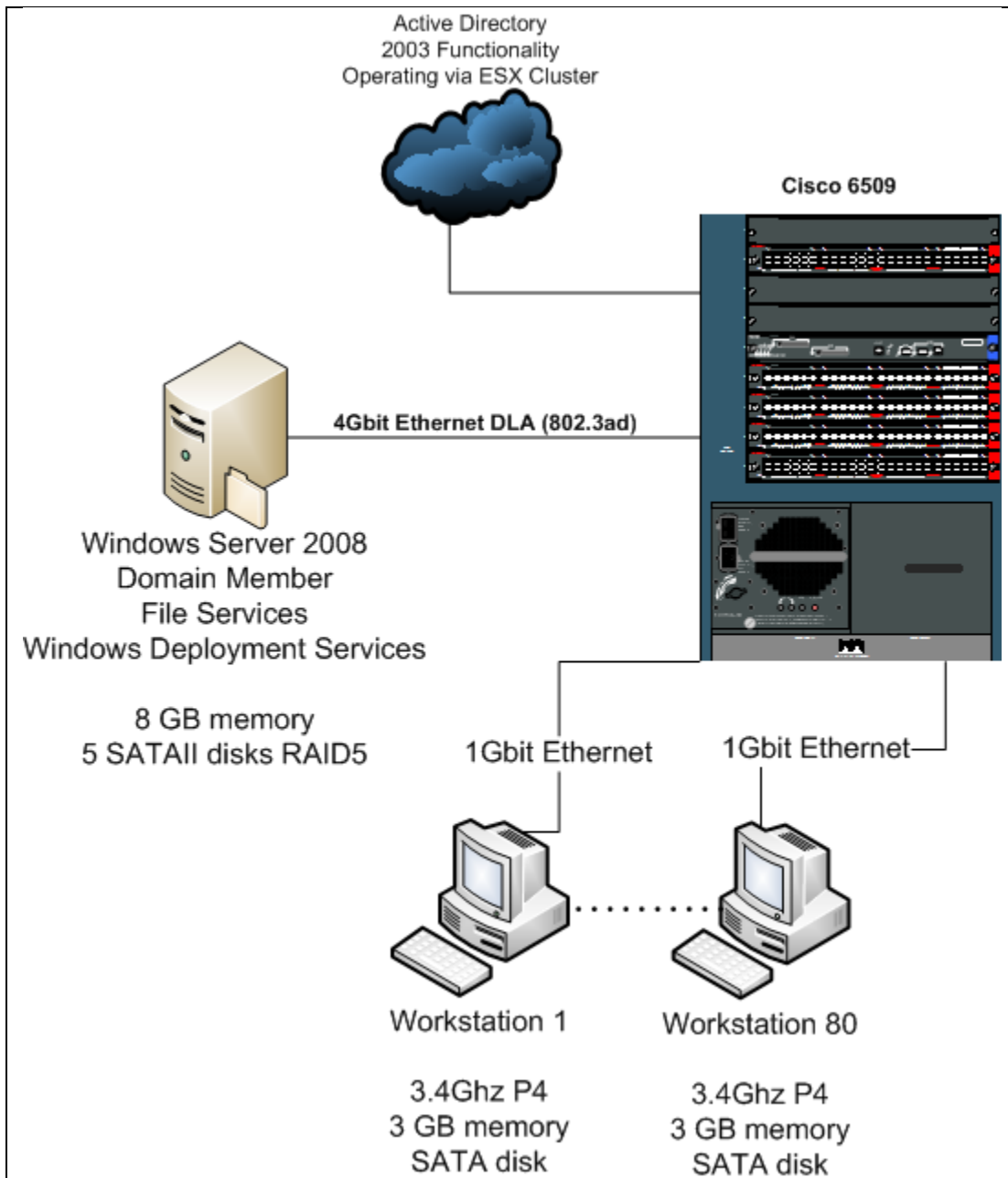

Figure 1 – Sample lab diagram

3.2 Environment Overview

This overview describes, explains, and justifies the procedures required to implement an environment that supports differential, workstation-based virtualization for mobile computer system modeling. The remaining subsections describe an implementation of the environment.

The ideal environment stores differential virtual machines at a highly available storage solution for two reasons. The first, mobility, is a necessary feature for modeling environments when there are more users than workstations. The second reason is to uphold data redundancy – store important data in some way to decrease the risk of data loss or inaccessibility in the unfortunate event of infrastructure error or failure. This is not to say that linked clones must be stored at a highly available storage solution all of the time. In fact, this research concludes that caching linked clones on workstations prior to executing might be more feasible.

On the workstations, an operating system and template virtual machines are installed prior to user interaction. Once a user has an authorized account and networked home directory, they can initiate logon sessions with at least one workstation. Once the user has initiated at least one session, they can create linked clones within their networked home directory. Once the linked clones are created, they can be configured and powered on — this can be considered the start of a user's modeling session. As the user manipulates their linked clones, their modifications are stored to the network share whenever the virtualization software dictates writes to the virtual disk or virtual memory. At any time, the user can snapshot, suspend or power off their linked clone to save the current state of their model to the file server — this can be considered the end of a user's modeling session. Once the modeling session is complete, the user can end their logon session. Later, when the user returns, they initiate at least one logon session to restart their modeling session by opening, configuring and powering on their linked clones.

3.2.1 Network Services

In chapter 2, the usefulness of network features like authentication and access control are presented. Using Microsoft Active Directory, Domain Name Services, and File Services, one can accomplish these features. However, this research environment does not require detailed configuration of these services beyond their basic operating state. Therefore, this research will not discuss their installation and configuration at length. It is pivotal to configure Active

Directory and Domain Name Services prior to other services because all remaining services rely on Domain Name Services to access Active Directory and further rely on Active Directory for authentication and role-based access control. This research recommends configuration of File Services prior to user creation.

The creation of users is an important and potentially time-consuming process. For this environment, it is important that each user obtains control of a home folder and that this home folder is automatically mapped when the user performs logon to a domain-joined workstation. The home folder is the central location where the user stores their linked clones. It is recommended to use a script to automate a majority of the process and prevent user-error. Such a script would likely utilize cacls.exe and icacls.exe to modify access lists on the home directories. Further, the directory services utilities are included in Server 2008 as the feature, RSAT-ADDS. The directory service utilities assist with managing AD objects and fan facilitate the user creation process. The add utility, dsadd.exe, enables object attribute configuration upon creation. When dsadd.exe is used in context of user additions, the `hmdir` attribute signifies the home folder and the `hmdrv` attribute signifies to which local drive the `hmdir` will be mapped when the user logs onto a workstation. For example, the command in Figure 2 adds a user named `john` to the default users' organizational unit with a home directory of `\\server\share\`. The share will be mapped to Z:\ when john logs on to a domain-joined workstation. In addition to the directory service tools, many Microsoft programming and scripting languages offer façades to the Active Directory Services Interfaces (Microsoft Corporation 2008g).

```
dsadd user "CN=john,CN=Users,DC=koppe,DC=thesis"
-hmdir \\server\share
-hmdrv z:
```
Figure 2 – Home directory specification upon user creation
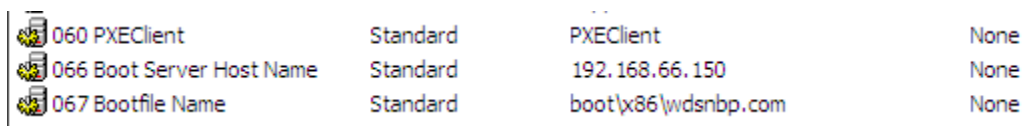
3.2.2 Workstation Deployment

The Dynamic Host Configuration Protocol, DHCP, leases an IP address to a computer that requests an IP address. For this thesis, it is assumed that DHCP is already in place and that administrators install Windows Deployment Services, WDS, on a separate server. This section begins by detailing what modifications to the DHCP service configuration are necessary to cooperate with a Windows Deployment Service instance executing on a distinct IP address.

Next, this section details the configuration of a WDS server to enable deploying operating systems to workstations.

There are many important DHCP configuration concepts. Since DHCP leases IP addresses, DHCP traffic does not natively traverse networks. However, DHCP relay agents perform this multi-network action. Further, DHCP has a notion of lease scope. This means that administrators must configure the DHCP service to offer and acknowledge a range of addresses. For the purposes of this thesis, network relaying and scope configuration is not of interest, but is required. For network usability purposes, practical implementations need to consider these configuration areas. For the purposes of illustrating how DHCP interoperates with WDS, configuration is performed on an entire DHCP server scope; however, one could extend these configuration steps to smaller scopes and more servers. Using the netsh utility (Microsoft Corporation 2005), one could dynamically configure DHCP settings at a specific time or through user-initiated script. For example, an administrator might only want WDS to function during a specific period – one way to achieve this is through automated, scheduled netsh tasks.

WDS requires three specific DHCP configuration settings including the address of the WDS server, a PXE specification, and the boot filename (Microsoft Corporation 2008c). The following screenshot, Figure 3, shows the specific DHCP options configured to direct PXE clients to download a specific boot file (boot\x86\wdsnbp.com) from a specific WDS server (192.168.66.150).

060 PXEClient	Standard	PXEClient	None
066 Boot Server Host Name	Standard	192.168.66.150	None
067 Bootfile Name	Standard	boot\x86\wdsnbp.com	None

Figure 3 – DHCP options required for WDS

Statically configuring the boot\x86\wdsnbp.com does not limit deployment to x86 architectures; architecture detection commences after the boot program executes. Further, since DHCP is on a separate server from WDS, the administrator must create and define option 60 using either netsh or the graphical interface. Figure 4 below shows the netsh commands to create and configure option 60.

```
netsh Dhcp Server Add Optiondef 60 "PXEClient" STRING 0 comment="PXE
```

```
Support" "PXEClient"
netsh Dhcp Server set optionvalue 60 STRING "PXEClient"
```
Figure 4 – Netsh commands to configure DHCP option 60

Now that DHCP directs PXE clients to the WDS server, the WDS server needs to be configured. The Windows Deployment Services (WDS) "enables rapid deployment of Windows to computers via network-based installation" (Microsoft Corporation 2008c). This research uses WDS to install the workstation operating systems. Since it offers multicast support, the version of WDS that this research uses is a role in Windows Server 2008. Use servermanagercmd.exe, as illustrated below in Figure 5, or the Server Manager graphical console to install the role.

```
servermanagercmd -install WDS
```
Figure 5 – Servermanagercmd to install WDS

Once installed, the server requires initialization. Initialization uses servermanagercmd.exe or the Server manager graphical console. The initialization process includes the creation of the Remote Installation share, DHCP option configuration, and PXE response configuration. Microsoft recommends placing the Remote Installation share on a different volume than the operating system. This research assumes the DHCP server exists at a separate address; therefore, the default DHCP configuration is acceptable. Finally, set PXE to respond to all unknown and known clients. The value of this configuration instructs the WDS server to permit access to specific workstations. Since the PXE response setting is an access control, consider security when configuring this in a practical environment.

Upon initialization, WDS suggests adding images. In WDS, there are different types of images; however, this research details usage of three types: install, boot setup, and boot capture images. The initialization wizard asks for an image source directory such as one found on Vista SP1 or Server 2008 media. By providing the sources directory, one boot setup image and some install images are added. A boot setup image enables the installation of an install image. For example, a workstation boots from the network and loads the boot setup image. Then, the boot setup image installs and configures a custom install image to the workstation.

To deploy custom images to the workstations in the modeling environment, an administrator can make a custom install image through a process known as capturing. First, a capture image is created on the WDS server. A capture image is created using wdsutil.exe, as illustrated in Figure 6 below, or the Windows Deployment Services console.

```
wdsutil /new-captureimage
/image:"Microsoft Windows Longhorn Setup (x86)" /architecture:x86
/destinationimage /filepath:"c:\capture.wim"

wdsutil /add-image /imagefile:"c:\capture.wim" /Imagetype:boot
/Name:"Microsoft Windows Capture (x86)"
```
Figure 6 – Creating and adding a capture boot image

The capturing process is similar to a traditional system imaging process. An operating system is installed on a reference computer and customizations are made. Then, the system must be prepared with the Microsoft Sysprep utility (Microsoft Corporation 2008c). While the Sysprep utility is included in Vista, it must be downloaded for XP as a part of the Deployment Tools package (Corporation 2008).

```
%systemroot%\system32\sysprep\sysprep /oobe /generalize /reboot
```
Figure 7 – Vista sysprep command to prepare a system for capture

```
sysprep –mini –reseal –reboot
```
Figure 8 – XP sysprep command to prepare a system for capture

Once prepared, the workstation reboots and begins the PXE process. To arrive at the boot selection screen, press F12 as directed in Figure 9 below.

Figure 9 – PXE process requiring F12 to boot

After hitting F12, the customized operating system on the workstation can be captured by selecting the capture image from the network boot screen as depicted in Figure 10 below. Note that the capture image will be displayed with the name specified when the image was added (see Figure 6 on page 17).
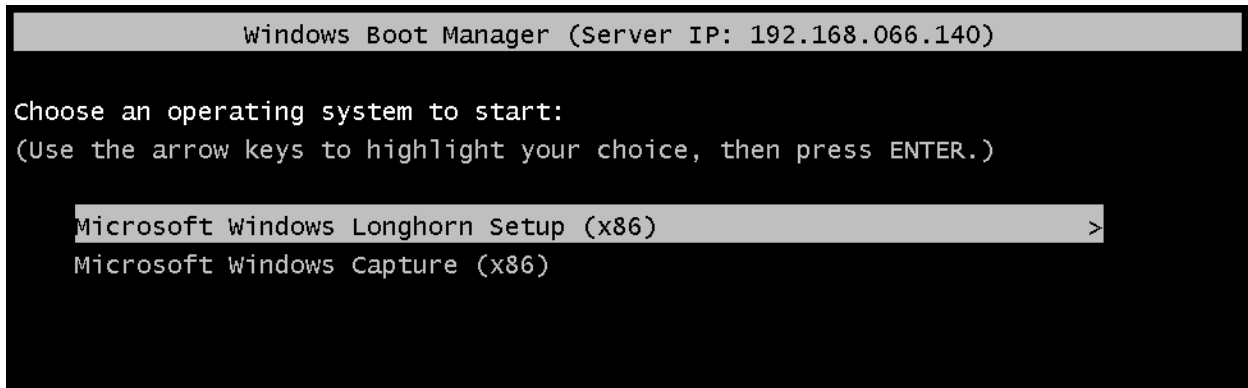
Figure 10 – WDS Boot Manager

WDS first captures the system image to a drive local to the workstation; therefore, there must be a volume with sufficient free space attached to the workstation prior to capturing. The next figures, Figure 11, Figure 12, and Figure 13, illustrate the capture process. Note that in Figure 12 the Location must have sufficient free space to store the capture image. In this example, C:\ is a 20GB volume with 4GB of used space. This means that the image will be approximately 4GB. Since there are 16 GB of free space on the volume, the image can be created successfully before being uploaded to the WDS server.
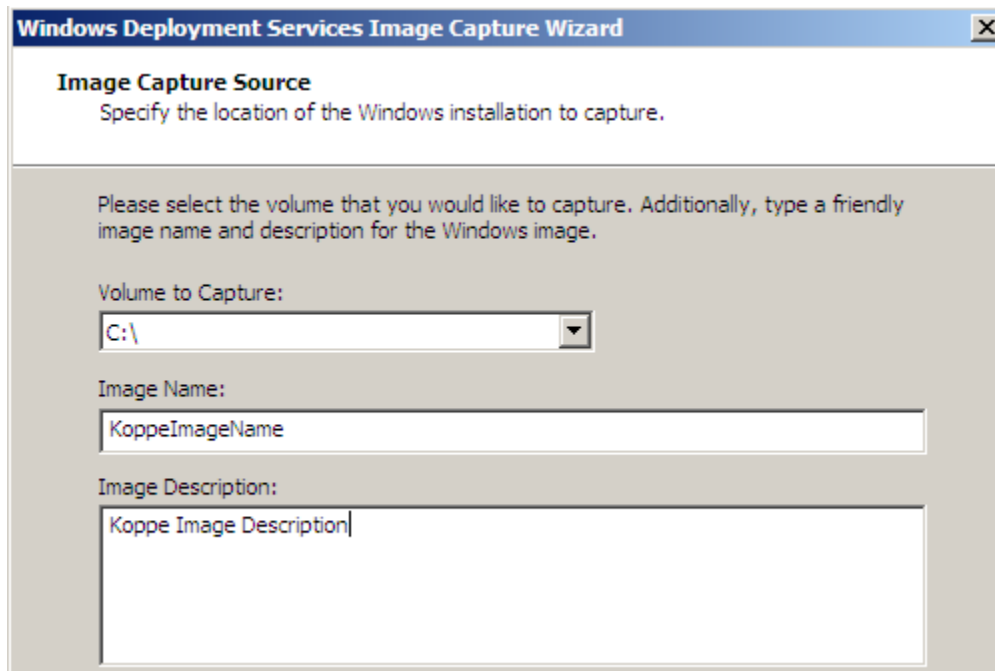
Figure 11 – Capture Wizard, Capture Source
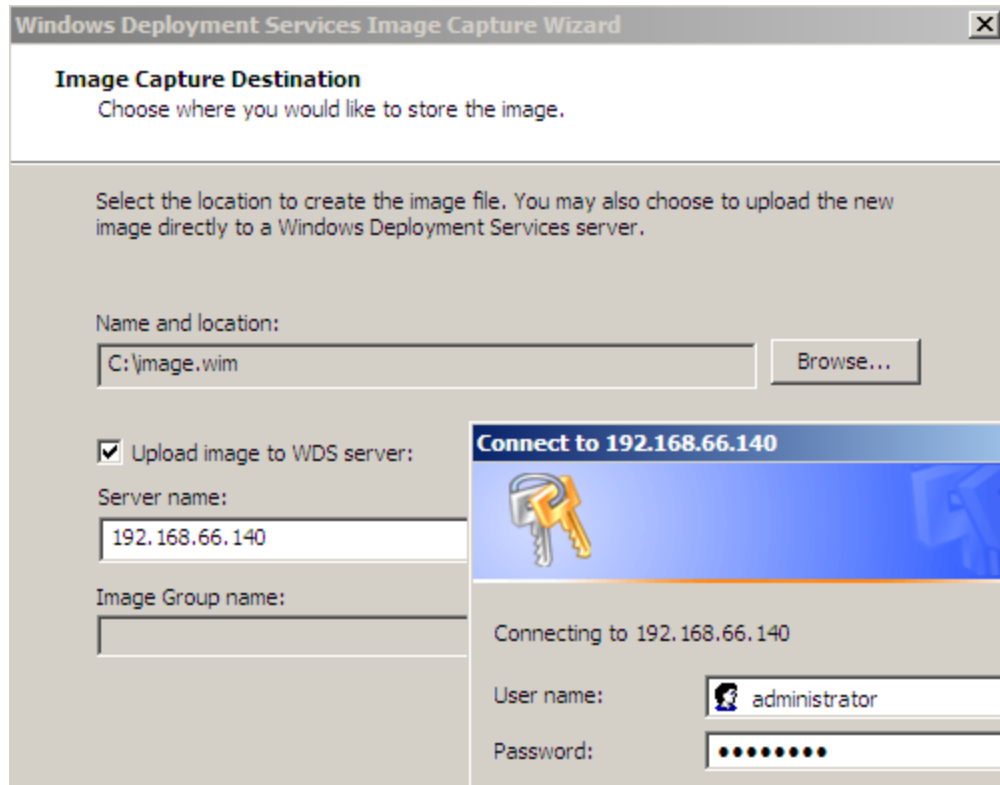
Figure 12 – Capture Wizard, Capture Destination
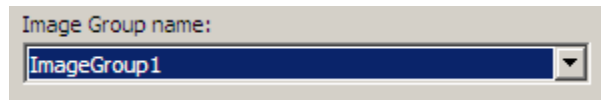
Figure 13 – Capture Wizard, group choice after authentication

After being captured to the local system, the custom install image is added to the WDS server and can be deployed to one or many workstations. This deployment process includes booting from the network, selecting a boot image, authenticating identity, selecting an install image, installing Windows, renaming the workstation, and joining the Active Directory domain. The deployment process can be performed manually or automatically. Manual deployment might take place when deployment is required for a small number of workstations. A manual installation for a small number of workstations is not time consuming and is similar to the capture process. This research suggests scripted multicast deployment of a specific install image to workstations using the massImage.ps1. This script automates the WDS configuration steps necessary to multicast an install image. The script requires PowerShell 2.0 CTP (Microsoft Corporation 2007b) and a default.xml similar to the one in appendix 7.6 on page 58. The

deployment process and other WDS-related processes are detailed in a Microsoft TechNet article, titled *Windows Deployment Services Processes* (Microsoft Corporation 2008d); further, the automated steps of the massImage.ps1 script are documented in the script heading comments (see appendix 7.5, page 51). An unattended installation can become very complex. The example in the appendix, default.xml, installs a Windows Vista image to a 20GB C:\ partition on the first disk, creates and formats a second partition P:\ with the remaining space on the first disk, renames the computer according to the WDS naming pattern, and joins the domain TESTDOMAIN with credentials TESTUSERNAME and TESTPASSWORD. In a true environment, care should be taken to assure that the TESTUSERNAME role has limited abilities; see *Performing Unattended Installations* from TechNet (Microsoft Corporation 2008e) for detailed information about automating installation.

In order for the workstations to name according to a naming pattern, they must either be prestaged or started in a specific order. WDS uses Active Directory to name computers. During the deployment process, WDS sets the workstation name based on the hardware UUID or the network adapter MAC address. If either of these is known, the workstations can be prestaged using wdsutil.exe (included when the WDS-RSAT feature is installed in Windows Server 2008). If not, WDS can assign names based on a defined pattern. For example, one could define a pattern `WS%02#` that would cause WDS to name new workstations WS01, WS02, WS03, etc. Ultimately, if workstation names are important to an organization, WDS offers flexible configurability to the workstation naming process.

3.2.3   Virtual Machine Templates

Virtual machine templates are the basis for linked clones. Configuring a template requires a few extra steps beyond basic virtual machine creation steps. Once the virtual machine is created, the virtual operating system in installed and customized, a snapshot must be taken. Once the snapshot is taken, the virtual machine must be configured to template mode. This is completed in the virtual machine settings dialog in the options tab as shown in Figure 14 below. Alternatively, a snapshot can be performed via the command line using the vmrun.exe utility and by adding `templateVM = "TRUE"` to the .vmx configuration file.
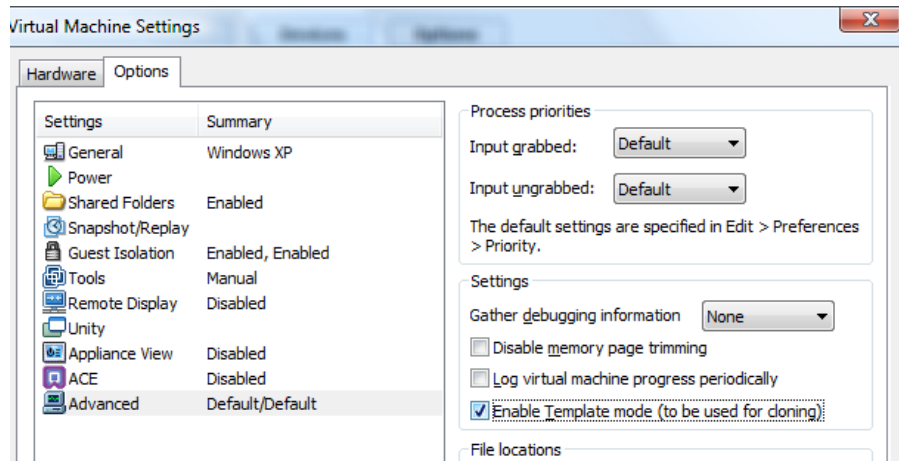
Figure 14 – Enabling template mode

Once the templates are created, they must be stored on each workstation. By storing virtual machine templates on each workstation, the environment offers a reduction in the time required for user model restoration (the experiment detailed in section 3.3 proves this claim). The task of restoring the base operating system or virtual machine template is removed from the user and given to management. The deployment process, as discussed in the previous section, is the opportune time to move virtual machine templates to the workstations. Prior to capturing the custom install image, the administrators could inject the virtual machine templates to a folder in the custom image.

After initial deployment, a process in which administrators can add, update, or replace virtual machine templates is ideal. If a new operating system is to be modeled or a template is configured improperly, the templates need to be updated on each workstation. There are a number of ways to achieve this. One way is to re-deploy the operating systems with an updated image. This is, however, time consuming and inefficient. It is further effective to copy the template modifications to each workstation instead of the entire operating system or all of templates. Using Robocopy.exe, an administrator could script copying just the differences between a virtual machine template repository and the template folder on each workstation (Microsoft Corporation 2008a). Undoubtedly, other third party utilities and file transfer protocols exist to perform such differential copying. However, they all perform the copying in parallel or series – unless they use multicasting. WDS offers the ability to multicast data to a workstation outside of an installation process. This is accomplished using a custom namespace,

created with `wdsutil.exe`, and having workstations join namespace with `wdsmcast.exe`. Workstations can join the namespace while executing their normal operating system if Windows Vista SP1 and Server 2008 AIK is installed (Microsoft Corporation 2008b). Otherwise, a custom boot image can be created on any server where the latest AIK is installed. The process of creating the custom image is detailed in a TechNet article titled *Using Transport Server* (Microsoft Corporation 2008f) in the section titled *Using a Transport Server for Multicasting*. A multicast namespace is created in much the same fashion that a multicast transmission is created. A script similar to massImage.ps1 could be created to facilitate the process of multicasting updated templates to all modeling workstations. Firstly, a template image must be created because multicasting is optimized for single file transfers (Sadler 2007). A template image can by making a differential-update version of the template repository in a folder using `Robocopy.exe`. Then, that directory must be mounted to a volume letter using `subst.exe`. Finally, the virtual disk can be captured using `imagex.exe`, from the Windows Server 2008 AIK. Once the template image is made, the image should be stored in the WDS REMINST subdirectory images. Then, the namespace must be created using the path to the images subdirectory as the `/configstring` parameter in a `wdsutil.exe /new-namespace` command. If using a custom boot image, the image should have `wdsmcast.exe, imagex.exe, wimfltr.sys, and wimfltr.inf` along with a startup script; these files come with Windows Server 2008 AIK. There is an example Win PE startup script, startnet.cmd, in appendix 7.8, on page 68. However, this method might require the workstation to have twice as much free space as the change in size of the template repository if additions or updates are required. For example, if an administrator added ten templates to the template repository and the repository size increases by 15GB to a total of 75GB. Then, if the hard disk in each workstation has 60GB of templates and 20GB of free space, this solution will not work. This solution requires that the workstation have 15GB of free space for the image to be stored by wdsmcast.exe and then requires an additional 15GB of free space for the extraction of the templates. However, if there is sufficient temporary space, this update method is network-optimized because of the multicast transmission and size-optimized because of the differential image. There will always be a time-tradeoff to analyze when choosing a template update process. If the update is small, it might be worthwhile to Robocopy in series rather than using the wdsmcast.exe approach.

3.2.4 Using the Environment

When a user has a valid account on the domain, they can begin to use the modeling environment. A user can perform three basic tasks: create linked clones, use linked clones, store linked clones. Thus, a user must be trained in linked clone management. A high-level usage perspective is that users create and use linked clones to model systems and processes. Users can store their linked clones on a centralized storage solution to retain a specific system state. Creating linked clones in VMware Workstation is straightforward: the user opens the template and selects Clone from the VM menu as shown in Figure 15. It is crucial that users understand they are to use linked clones and not full clones. In the cloning process, a user has the option to create a full clone. However, this defeats the purpose of differential virtualization and users must be educated.
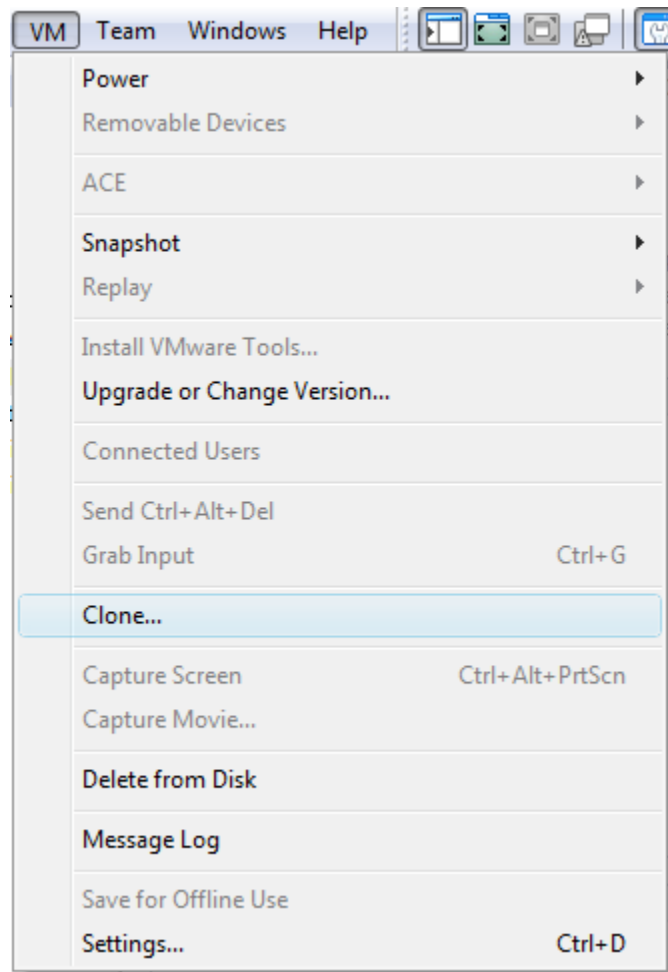
Figure 15 – Clone selection

The user then selects the template's snapshot and the linked clone option. Lastly, the user must specify a name and location to save the linked clone. There are two optional locations for linked clones: the workstation or the file server. Linked clones should be stored in specific locations based on their state. Linked clones exist in either the on or off state. When off and not active in a model, linked clones should be stored on the file server in order to uphold the mobility requirement of the shared modeling environment. However, during linked clone use, location is a critical to usability. As detailed in the experiment in section 3.4, the computing resources required for direct modification of linked clones on a file server exceed those offered by the sampled file server. Thus, location during use must carefully be considered based upon file server capabilities.

If the file server is capable of sustaining reasonable disk throughput when many linked clones are being modified directly from the file server, this is the optimal modification approach. When users save, open, and use their linked clones directly from the file server, this is called the direct modification approach. However, if the file server is not capable of sustaining sufficient disk throughput, the users must save and use their linked clones on each workstation. It is possible to facilitate caching of linked clones on workstations using Windows Offline files or other third party applications; however, it is not difficult for users to copy linked clones selectively to their desktop for use in a modeling session. This approach can be considered a cache-and-update approach. The linked clones must be cached to the workstation before the modeling session, executed on the workstation during the modeling session, and updated on the file server after the modeling session. The other approach, where linked clones are stored and used directly from the file server is called the direct modification approach.

Once the user has created, and possibly cached, the linked clone, using VMware Workstation as their personalized modeling environment is nearly as straightforward as using a traditionally hardware-specific system. See the VMware Workstation documentation for usage instructions (VMware, Inc. 2008b).

3.3   Optimizing Restoration Time

As discussed earlier, users perform basic tasks in the modeling environment: create, use, save, and restore models. Consider restoration time as the amount of time it takes a user to restore a

24

modeling environment from a stored state. This process includes the workstation startup, user login, downloading their stored model, and finally reaching a stage where their model is usable. Logically, by reducing the time it takes to restore a model, the user has more time to manipulate and work with the model.

The purpose of this experiment is to determine the amount of time it takes to restore many modeling environments using a traditional restoration technique as well as how long it takes to restore similar modeling environments using two variants of the differential virtualization approach. By manipulating restoration approach and workstation operating system, one can determine how the dependent variable of restoration time is affected.

In this experiment, the model environments are pre-configured and saved for a number of simulated users ranging from one to sixty. The models are set to automatically login and execute a startup script. This script will write a file to the file server in the folder specific to the workstation. Therefore, by noting the time that the systems started and automatically creating a file when the model is operational, one can understand how the restoration time changes with respect to the number of model environments for each restoration approach and configuration. With intent to correlate component ability with restoration time, the performance of the file server is measured during restoration experiments; specifically, the disk operations per second, disk bytes per operation, disk queue length, and network bytes transferred per second.

There are many assumptions for this experiment. Restoration time is considered in this experiment; however, the time to save a model is similar to restoration time and save time can be assumed dependent on the configuration in the same manner as restoration time. It is assumed that there exists a pre-restoration process. This means, that the time it takes a user to situate themselves at a set of workstations on which they will restore their model is entirely a different stage of the modeling experience. It is assumed that there exists a post-restoration process. This means that the user must interact with the restored modeling environment in a usable fashion. While usability of the model plays a role in this experiment, it is not the primary intent. As users modify a system instance within a model, the size of the model can increase. It is assumed that restoration time increases somewhat linearly as users make more modifications to their models. To that end and for expediency, the model-specific modifications are limited in this experiment.

The Ghost images and linked clones used in this experiment are Windows 2003 Server based with no user-specific modifications other than the small scripts to automate the experiment for the purposes discussed in the previous section.

In order to automate the experiment, certain pre-experiment configurations are necessary. IP addresses are unique per physical workstation. A DHCP server assigns specified IP addresses to workstations based on the MAC address of the workstation's physical hardware. A unique domain user account exists for each workstation. Each domain user account has full access control of exactly one folder on the file server. In this folder, there exists a Ghost image and a linked clone. For the ease of this experiment, all domain users can read these folders. Next, the workstations are configured such that they first attempt to boot from a network location and then the local disk. The sample laboratory has eighty workstations and some are not available due to configuration errors or hardware issues. Therefore, at most sixty models are used in this experiment.

To illustrate how restoration time changes with different approaches and configurations, three heat sizes, one, thirty, and sixty, are designated. The size of the heat is representative of the number of unique systems modeled in the environment. Therefore, a heat with a size of sixty signifies the use of sixty workstations to achieve sixty unique modeled systems. This is analogous to sixty users each modeling a unique single operating system. Three restoration approaches are considered: Symantec Ghost system imaging, Windows Vista based cached differential virtualization, and Windows XP based cached differential virtualization. In all configurations, Windows Server 2008 is used as the operating system executing on the file server. Effectively, this experiment is a cross-sectional survey with nine resulting datasets.

The technical objective of the experiment is to have one result file per workstation per heat. This way, statistical analysis can be performed on a group of data points from a specific experiment. At the end of the heat, a script finds the result files, calculates the difference between the start of the test and the creation time of the result files, and generates a row in a results database for the each workstation. Each row contains the test type, heat size, workstation name, and restoration time in seconds. The following example database record signifies that a workstation named maul101 took approximately 8935 seconds (or, approximately 2.5 hours) to restore using the

Ghost imaging approach while 59 others simultaneously restored. A simple script, called batchanalyze.ps1, was written to perform basic data analysis a comma-delimited database using this schema. For longevity, the result database is included in appendix 7.1, on page 44.

```
traditional,60,maul101,8935.3245675
```
Figure 16 – Restoration experiment result record example

At the beginning of the test, each workstation is pre-configured and powered off. To start the test, a script called starttest.ps1 is executed on the file server. This script gathers input regarding the test type and heat size. Upon input, the script notes the time and in quick succession, the specified number of workstations are manually powered on. Starting the workstations will continue down one of three avenues:

1. Automatic imaging using Symantec ghost
2. Windows Vista boot process
3. Windows XP boot process

The traditional preparation technique uses Symantec Ghost to restore a single operating system to a single physical machine. In order to automate the imaging process, a special PXE boot image was created. Since the workstations are configured to boot from the network, once the workstation starts, it downloads and executes this special PXE boot image. The special PXE boot image loads DOS, loads necessary drivers and maps a network folder based on the IP address of the workstation. Then, each workstation downloads a unique Ghost image from the file server to its local disk. It is crucial that each workstation reads a separate Ghost image in order to simulate a realistic restoration process. When two users restore their unique environment, they are reading a distinct set of files, not the same set. Therefore, each workstation will copy a unique Windows 2003 R2 image to their hard drive and reboot. On startup, a local user will automatically login and execute a script. This script will map a network drive based on the IP address of the workstation and write a file to that network drive as described earlier.

In the differential virtualization approaches, both Windows Vista and Windows XP are used as the workstation operating systems. In order to accomplish the test, the system images were prepared in a similar fashion as the images from the Ghost restoration approach. However, in

these tests, workstations are domain-joined and configured to auto-logon based upon the name of the workstation. When the workstation starts, the domain user specific to that workstation automatically performs the logon process and their network folder maps to Z:\. Then, a script empties the user's desktop, copies the linked clone from Z:\ to the user's desktop, and starts the linked clone. Further, once the linked clone starts, the same script runs a script inside the clone that writes the file to the workstation-specific folder on the file server.

3.4   Direct Modification Characterization

As described earlier, an optimal environment is one where linked clones are modified directly on a file server, not one that uses the cache-and-update approach. In the first heat of size sixty for differential virtualization, the direct modification approach was utilized. Unfortunately, it was quickly apparent that the tested file server would not support many simultaneously operating linked clone virtual machines. This section analyzes the requirements for an environment where direct modification is the primary approach when manipulating linked clones from a file server.

The purpose of the direct modification experiments is to determine requirements for executing linked clones from network storage services. Clearly, after discovering that the file server in used in the restoration experiment could not support sixty direct modifications, this behavior requires further research.

In the experiments outlined in section 3.3, network utilization never reached its theoretical upper limit. Even in the XP restoration experiment, when a maximum of 58,190,595 bytes per second transferred through the network adapter, the adapter only reached roughly 11% utilization of its 4Gbit bandwidth. It is assumed that the storage solution is a limiting hardware piece of the IO path. Since it is assumed that the storage solution is the problematic component in the IO path, the data gathering and analysis is limited to primarily statistics of the file server disk. Further, the model operating systems are limited to just Windows Sever 2003.

In this experiment, hardware performance statistics are gathered using Windows Performance Monitor while one or many virtual machines are created, opened, or closed using the direct modification approach. First, the restoration time experiment is attempted using direct modification with sixty models executing from Vista workstations and performance statistics are gathered. Then, in a separate test, a linked clone is created in a home folder and performance

statistics are gathered. Then, in another test, the linked clone is shutdown and performance statistics are gathered. Finally, in the last test, the linked clone is started for a second time, simulating a restoration, and the performance statistics are gathered. Thus, as a result, there are four Performance Monitor data sets. One set from which analysis will show how many attempts to utilize the direct modification approach affect the file server's resources. While, from the other three sets, analysis will show how specific actions on a single direct modification of a linked clone affect the file server's resources. Finally, a performance correlation is attempted between a single direct modification and many direct modifications.

3.5   Workstation Capabilities

Until now, experiments have focused on differential system modeling using a single model per workstation. However, virtualization plays a major role in this research because it facilitates the operation of multiple models per workstation. The purpose of this experiment is to determine the capabilities of workstations regarding the execution of virtual machines.

Firstly, it is assumed that users can intelligently manage resources allocated to each virtual machine. In this modeling environment, the user has full control of their virtual machines; therefore, the user must set reasonable resource limits for their virtual machines. The user must understand that the workstation has a finite amount of disposable resources and that the workstation operating system requires a portion of those resources. It is not assumed that there exist precise resource limits but rather that the user can gauge usability within a model on a per-model basis. For example, if a user runs a Windows Server 2003 virtual machine with 92MB of memory and they are unable to run more than a few programs inside the virtual machine, the user *should* gather that the virtual machine might benefit from an increase in virtual memory.

In this experiment, test is restricted to three model operating systems: Windows XP Professional, Windows 2003, and CentOS 5. The same sample laboratory from previous experiments is utilized. Virtual machine memory allocation remains at the default allocation size as set by VMware Workstation virtual machine for a specific operating system type is created. Further, the default VMware Workstation memory preferences are used.

This experiment is straightforward; using XP and Vista, multiple virtual machines are launched on one workstation and ensure that each virtual machine is responsive. Table 1 below details the amount of virtual memory allocated to each virtual machine for this experiment.

Virtual Machine	Memory Allocation
Windows Server 2003	384MB
Windows XP Professional	512MB
CentOS 5.1 Server	384MB

Table 1 – Memory allocation for virtual machines

4 Results and Analyses

The logical services required to support a multi-user, shared, workstation-based modeling environment greatly enhance the network from a management and usability standpoint. Administrators can easily update model templates and can restrict access on a user or role-based level. Users are able to move between workstations without worrying about hardware dependency. It is crucial for an administrator to understand the purpose and configuration intricacies each component and process. When compared to an environment that supports modeling via traditional full-system based imaging, this environment has the same essential services. The addition of Windows Deployment Services in the researched environment is a complex process to learn. However, it is no different from learning the complex traditional imaging process. In the researched environment, differential modeling reduces the amount of user-specific data; thus, the overall storage requirements are decreased. In both the traditional approach and the researched approach, a significant amount of managerial tasks exist including maintain workstation images, maintaining models, and maintaining the network services. In the researched approach, models are built once; whereas, in the traditional approach, the models are hardware-specific and must be recreated when hardware changes. For those reasons, management in the researched environment requires less attention than management in the traditional environment. The remainder of this chapter presents results and analysis for the three experiments described in chapter 3.

4.1 Optimizing Restoration Time

The imaging restoration approach costs more than two hours longer than the differential virtualization approach with a heat size of sixty. Figure 17 shows the average time, in seconds, that it took to restore one, thirty, and sixty modeling environments.

Simultaneous Environment Restoration

Number of environments	1	30	60
Imaging	1294.039	4230.85	8784.804
DiffVirt-XP	190.184	208.911	322.306
DiffVirt-Vista	288.51	221.059	286.37
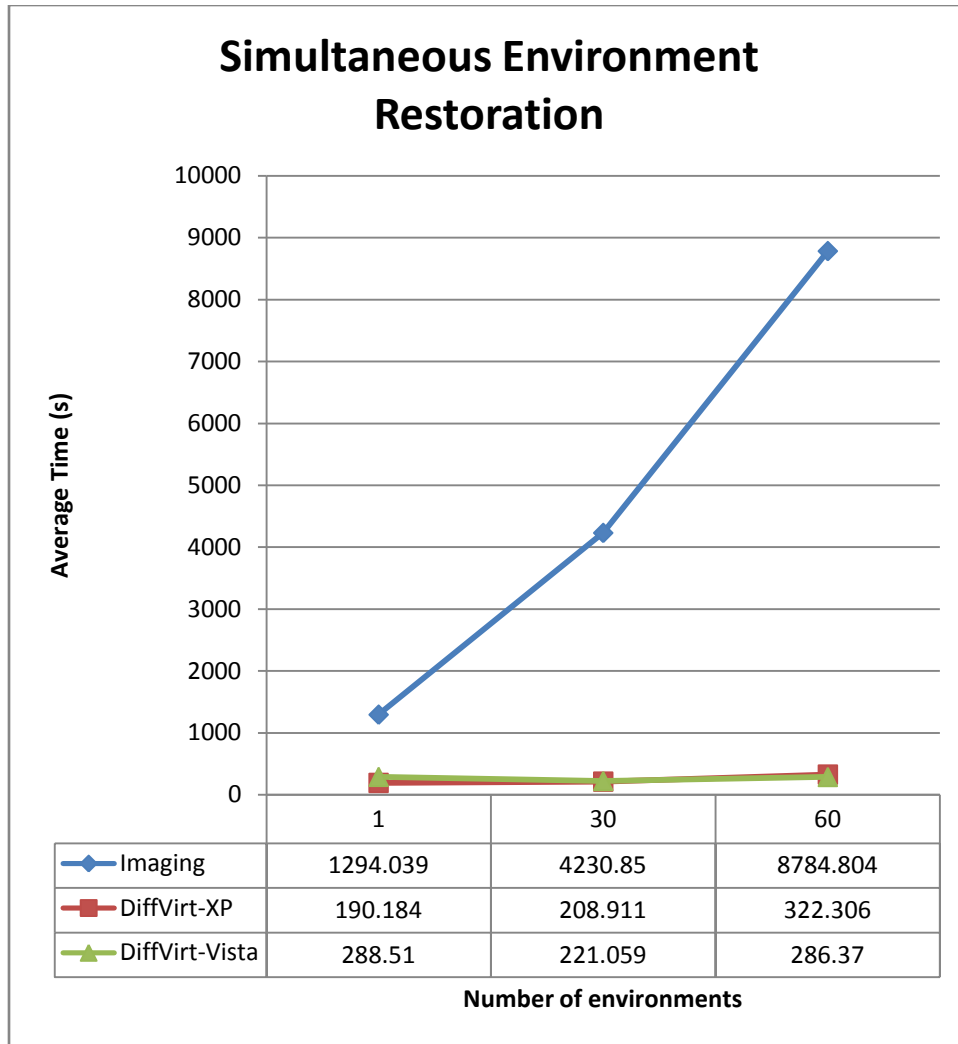
Figure 17 – Simultaneous restoration of models

While the experiment seeks a specific number of result files, this goal was not met during each heat. For example, in the Vista based restoration approach, less than all of the linked clones properly executed their startup script. The logon script used in the Vista test was not as effective as the script used in the XP test. The script used in the XP test, diffVirtResult.ps1 in appendix 7.2 on page 48, passes the workstation name as a parameter to the `vmrun.exe runprograminguest` command; whereas the script used in the Vista test utilized `renamefileinguest` and then `runprograminguest`. The `vmrun.exe renamefileinguest` command did not consistently function. Therefore, the Vista test yields a less accurate representation of restoration time than the XP test and the remainder of analysis will focus on describing the differences between the imaging and XP approaches. Even though the actual

number of result files does not equal the number of expected files; the expected number of linked clones successfully restored and started. Table 2 details the actual number of result files generated for each heat. The Vista based differential virtualization test has significantly lower actual reports in the thirty- and sixty-sized heats whereas the imaging tests have slightly lower actual reports.

Expected	30	60
Actual Imaging	29	57
Actual DiffVirt-XP	30	60
Actual DiffVirt-Vista	20	31

Table 2 – Expected versus actual reports from restoration experiments

Statistics were gathered regarding the server hardware performance using the Windows Performance Monitor during the restoration tests for a ten-minute period. In Table 3, the relevant statistics gathered during the XP restoration test with sixty modeled systems are shown. In Table 4, the statistics gathered during the imaging restoration test with sixty modeled systems are shown.

Item (per sec)	MAX	AVG	TOTAL	%	SUM
Disk Reads	866	560	110,336	99.96	110,385
Disk Writes	4	0	49	0.04	
Disk Read Bytes	55,836,846	30,528,474	6,014,412,431	99.99	**6,014,756,007**
Disk Write Bytes	28,673	1,744	343,576	0.01	
Disk Queue Length	39	20.9949			
Network Sent Bytes	58,190,595	25,888,362	2,200,510,747	98.87	2,225,584,540
Network Received Bytes	900,447	288,205	25,073,794	1.13	

Table 3 – Performance statistics during the XP restoration with sixty models

Item (per sec)	MAX	AVG	TOTAL	%	SUM
Disk Reads	1,000	721	431,661	100.0	431,661
Disk Writes	-	-	-	0.00	
Disk Read Bytes	32,450,628	23,350,984	13,987,239,500	100.0	**13,987,239,500**
Disk Write Bytes	-	-	-	0.00	
Disk Queue Length	18	15.2733	-	-	-
Network Sent Bytes	33,759,377	25,898,157	3,599,843,808	91.05	3,953,833,288
Network Received Bytes	3,275,002	2,510,564	353,989,479	8.95	

Table 4 – Performance statistics during the imaging restoration with sixty models

Table 4 represents the data from a ten-minute period during active download of the system image from the file server to the workstation disk, whereas Table 3 represents the data from a ten-

minute period containing the entire test. Recall that the XP restoration test with sixty models took less than ten minutes in its entirety. Therefore, just the relevant data for the XP restoration test from the period where the file server had disk activity, 0:57 – 4:13, is of interest. The following two figures, Figure 18 and Figure 19, show, in line charts, the disk statistics for file server during the XP restoration and imaging restoration experiments, respectively.
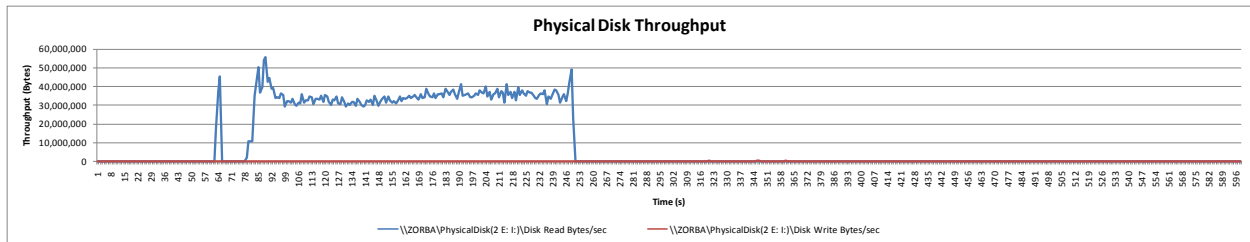
Figure 18 – Disk throughput during XP restoration with sixty models
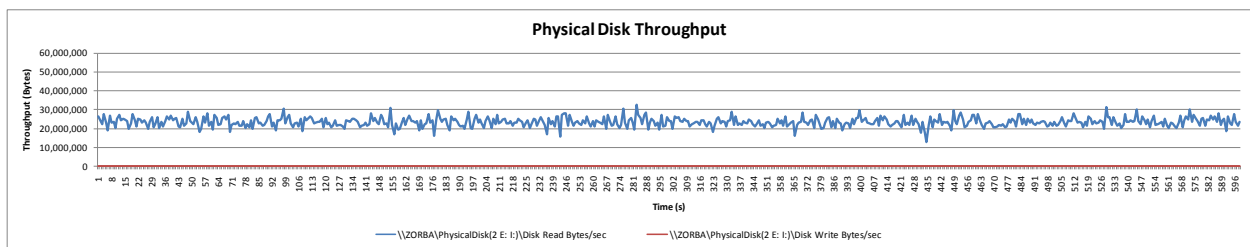
Figure 19 – Disk throughput during imaging restoration with sixty models

It is easy to categorize three stages of the XP restoration experiment when the throughput is represented in this fashion. The first minute, the workstations are powering on. For the four minutes where disk reads are apparent, the workstations copy linked clones to their local disk. Finally, the last half of the chart shows no activity – there is no interaction with the file server once the workstation caches its linked clone. The charted imaging throughput statistics shows a consistent amount of disk activity on the file server during the entire ten-minute gathering period. Further, one can also see that the disk does not deliver as high of a throughput during the imaging restoration as it does in the XP restoration.

Since linked clones are, by definition, much smaller than full system images, the amount of unique, yet similar, data transferred during the imaging restoration experiment considerably larger. The following two figures, Figure 20 and Figure 21, show the sizes of eighty linked clones and eighty Ghost images, respectively. Note that in this instance, each linked clone is

approximately 0.091GB (93MB) versus the 3GB of a Ghost image. Recall that these models are only slightly modified from the model template – modifications consist of installing scripts to automate the test. The differential nature of linked clones is the contributor to the stark size difference in these models.
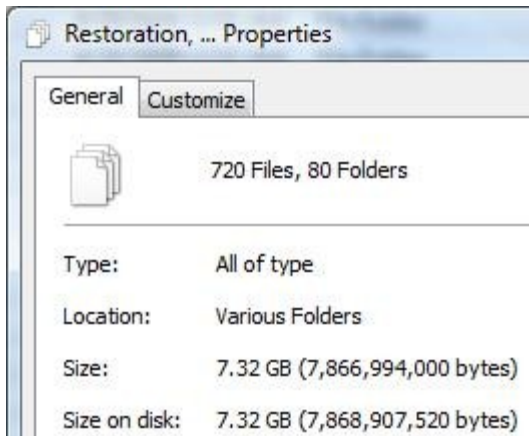
Figure 20 – Size of eighty distinct linked clones
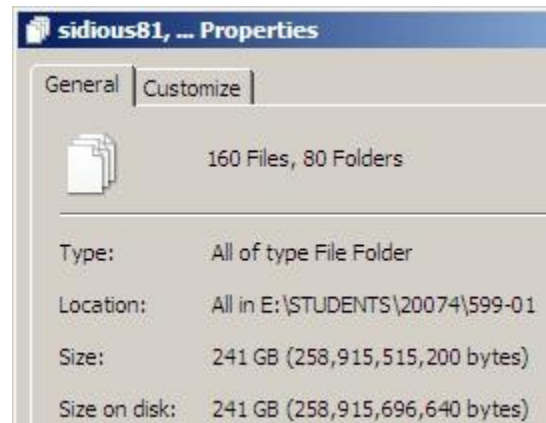
Figure 21 – Size of eighty distinct Ghost image

Therefore, by using cache-and-update differential virtualization, less time is required for the model restoration phase. Further, given an environment comparable to the sampled environment, a cache-and-update restoration and archival approach consistently yields high throughput as the number of models increases. In restoration, when caching is occurring, the disk throughput is characterized by nearly all read operations.

4.2 Direct Modification Characterization

Firstly, the direct modification approach to restoration in this experiment uses the same experimental design from section 3.3. In this instance, however, only eighteen models reported in the ten minutes, with an average restoration time of approximately 371 seconds (this data is in the Restoration Result Database in appendix 7.1 on page 44 as diffvirt-vista,70-17). After ten minutes, the usability of the "operational" models was very low – one could not even click a button inside the model because the virtual mouse would not respond. Since the high-level requirement of usability did not exist, perhaps low-level performance measurements reveal a cause for low usability. The performance data gathered for a ten-minute period during a Vista restoration test with sixty modeled systems using the direct modification approach are represented in Table 5.

Item (per sec)	MAX	AVG	TOTAL	%	SUM
Disk Reads	197	75	45,151	43.97	102,678
Disk Writes	139	96	57,527	56.03	
Disk Read Bytes	3,559,645	1,043,720	625,188,450	49.86	**1,253,883,296**
Disk Write Bytes	2,481,341	1,049,574	628,694,846	50.14	
Disk Queue Length	18	10.2300			
Network Sent Bytes	3,066,858	1,552,146	215,748,251	66.48	324,536,118
Network Received Bytes	1,351,262	771,545	108,787,866	33.52	

Table 5 – Performance statistics during the direct modification restoration with sixty models

The sum of disk throughput, at approximately 1.2 billion bytes, for the ten-minute period described in Table 5 is much lower than the sum of the disk throughput, at approximately 6 billion bytes, in the XP cached restoration approach (see Table 3, page 33). But, why? One possibility is that the file server cannot handle reading and writing operations in this capacity since the direct modification approach, in this capacity, has a read-to-write ratio of nearly one, whereas in the XP cached restoration approach it was nearly 100 (see Table 3, page 33). Another reason is that maybe the file server hardware cannot handle writing in this capacity. Figure 22, below, helps us to compare the read versus write operation throughput during the ten-minute direct modification test with sixty models by showing the percentage of disk reads or writes per second. The bottom area (blue) represents the read bytes per second while the top area (red) represents the write bytes per second. One can loosely see that more the linked clones read more bytes in the beginning of the ten-minute period whereas, in the end, they wrote more bytes.
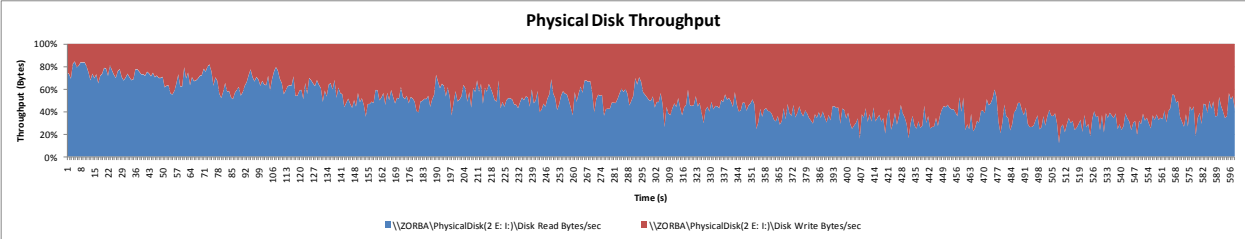
Figure 22 – Disk throughput during direct modification restoration with sixty models

Thus, one might gather that while read throughput exceeds the write throughput, linked clones more easily operate directly from the file server. Yet, when the write throughput exceeds the read throughput, the total disk throughput decreases causing linked clone usability and operation to diminish. However, it is likely that this analysis is sample-specific. One cannot easily extrapolate this characteristic to other storage systems without similar testing utilizing the other hardware.

In an effort to understand requirements that a file server needs to meet in order to enable direct modifications of linked clones located on the file server, the following three sections detail the disk throughput from the file server during three distinct linked clone operations.

A linked clone begins as a few files whose combined size is less than one megabyte, because, initially, a linked clone is just a set of pointers to the template virtual machine. The following graphs, Figure 23 and Figure 24, show the disk operations at the file server when starting a linked clone for the first time from an XP and Vista workstation, respectively.
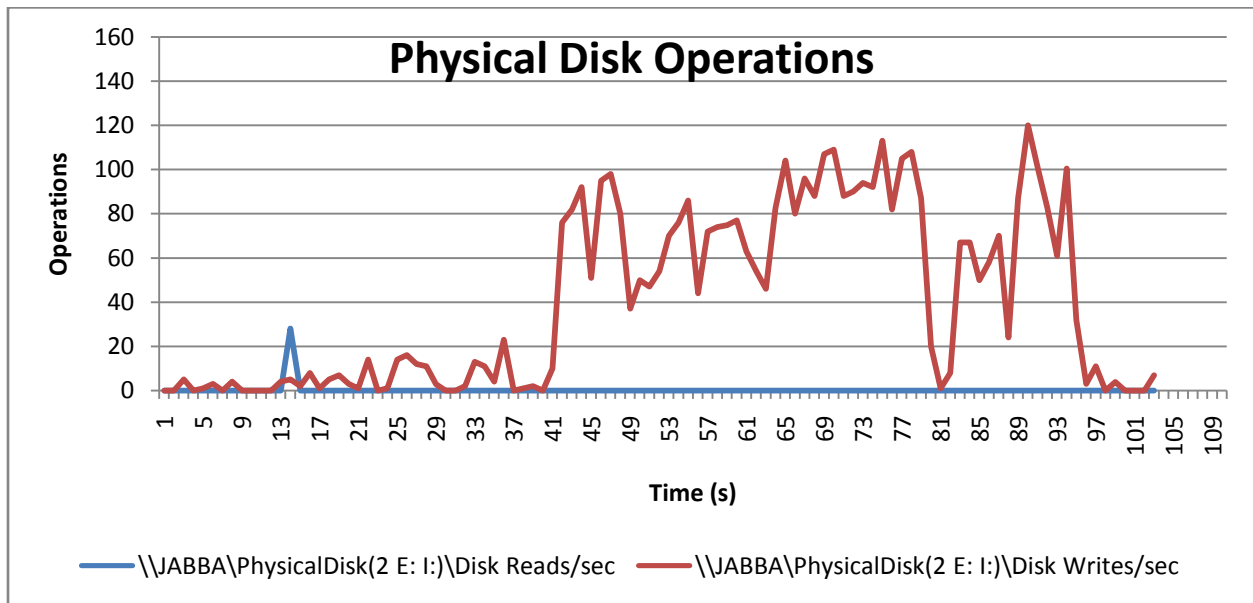
Figure 23 – Disk operations during first launch of model using XP direct modification
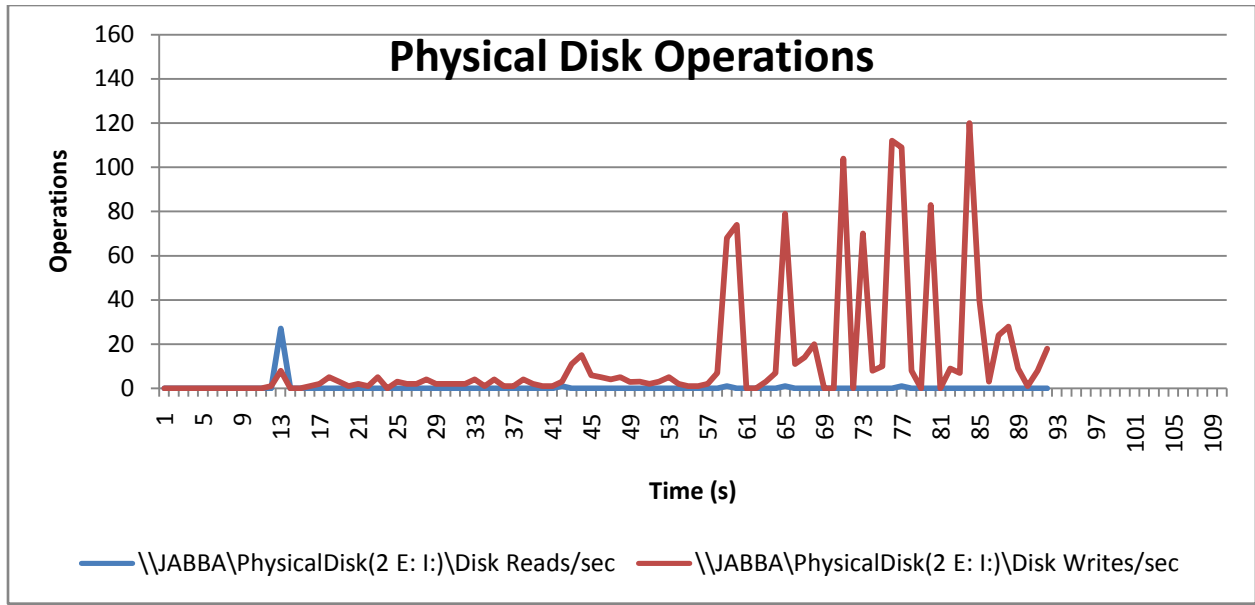
Figure 24 – Disk operations during first launch of model using Vista direct modification

The table below, Table 6, shows the read-to-write operations ratio for an initial launch of a linked clone directly from a file server using both XP and Vista as workstation operating systems.

Workstation OS	Reads	Writes	Read %	Write %
XP	28	4185	*0.7*	*99.3*
Vista	31	1192	*2.6*	*97.4*

Table 6 – Disk operations from server during initial launch of one model

Given these two data representations, one can understand that the initial execution of a linked clone from a networked file server primarily consists of *write* operations.

For this next experiment, a Windows Server 2003 linked clone was prepared by promoting it to a domain controller in a new domain. The modified Windows Server linked clone was then restored using the direct modification approach. The following graphs, Figure 25 and Figure 26, show the file server disk throughput when starting the modified linked clone from XP and Vista workstations, respectively.
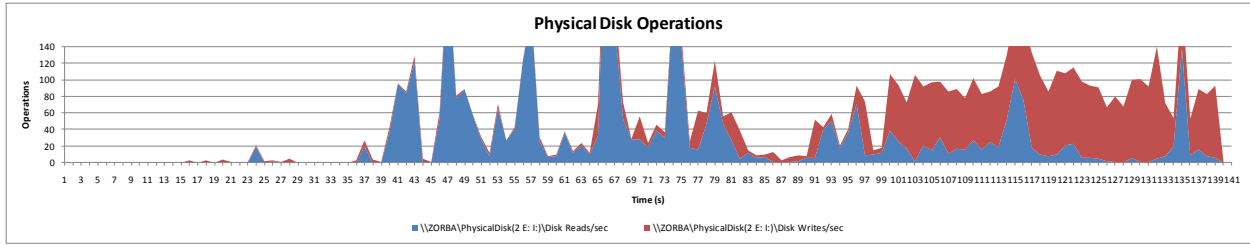
Figure 25 – Disk operations during launch of modified model using XP direct modification
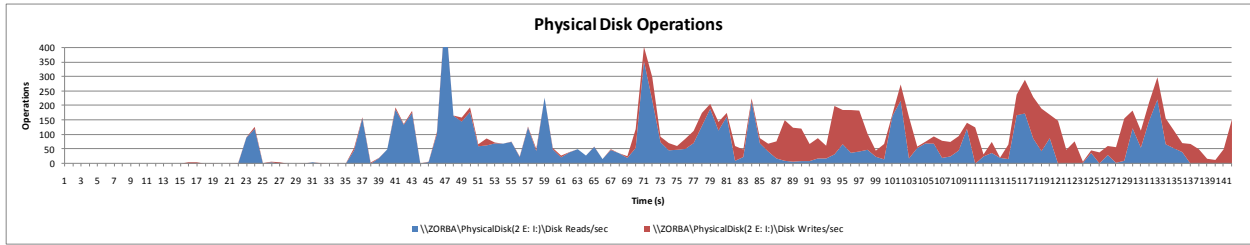
Figure 26 – Disk operations during launch of modified model using Vista direct modification

In both graphs, the linked clone starts at approximately twenty seconds (near the first spike). The top area (red) represents writes per second while the bottom area (blue) represents reads per second. Both graphs appear to have similar trends of throughput with respect to time. Both graphs appear to have more reads than writes early on, however this behavior seems to switch towards the end of the launch. The table below, Table 7, illustrates the disk operations for the launch of the modified linked clone, a period that was approximately 140 seconds in both cases.

Workstation OS	Reads	Writes	Read %	Write %
XP	3,940	3,804	*50.88*	*49.12*
Vista	7,699	4,598	*62.61*	*37.39*

Table 7 – Disk operations from server during launch of one modified model

Given these two data representations, one can understand that the launch of a previously modified linked clone from a networked file server consists heavily of both *read and write* operations.

In the final direct modification experiment, statistics were gathered while linked clone was being shutdown. The following representations, Figure 27, Figure 28, and Table 7, illustrate disk throughput on the file server during the shutdown of the linked clone. In both figures, the bottom area (blue) represents read operations per second while the top area (red) represents write operations per second. The figures show similar trends with low amount of operations for the

39

first half of shutdown followed by a larger amount of operations toward the end. Further, the majority of the read operations occur in the second half of shutdown.
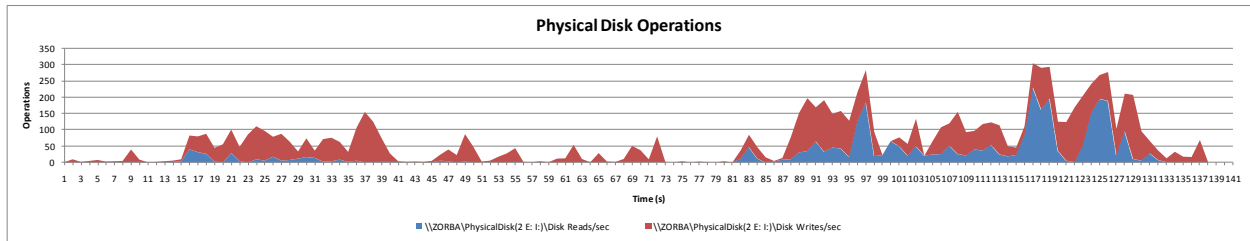
Figure 27 – Disk operations during shutdown of modified model using XP direct modification
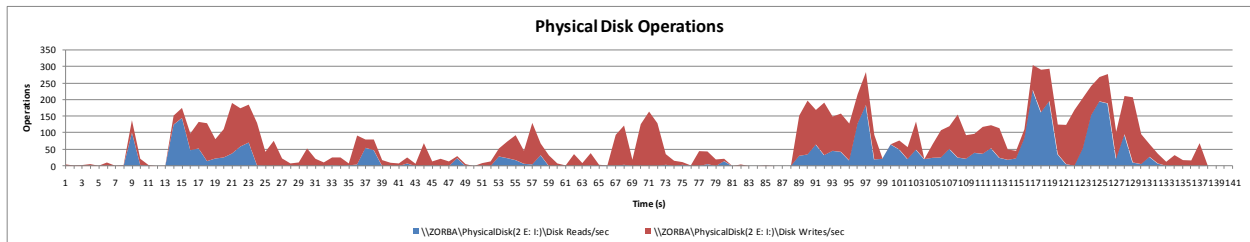
**Figure 28 – Disk operations during shutdown of modified model using Vista direct modification**

Workstation OS	Reads	Writes	Read %	Write %
XP	3,037	6,411	*32.14*	*67.86*
Vista	3,664	6,831	*34.91*	*65.09*

Table 8 – Disk operations during shutdown of one modified model

Given these representations, one can see that the shutdown of a linked clone on a networked file share consists of both read and write operations; however, more writes than reads comprise the shutdown process.

It was determined that by using a direct modification restoration and archival approach with an increasing number of models, throughput is greatly reduced. When one or many linked clones are restored directly, disk operations consist of a nearly equal amount of reads and writes. Thus, an argument could be made that when both types of operations occur simultaneously, a piece of hardware in the IO path is not capable of delivering the required throughput. In the instance of the researched environment, a single device type was not determined to be the root cause. To determine how network bandwidth on the file server affects restoration time, further experiments could decrease the allotted bandwidth of the network adapter on the file server and observe re-perform the restoration experiment. In the experiments, the total file size of a linked clone was at most 100MB. Therefore, the largest file in a linked clone file set is equal to or smaller than

100MB. Whereas, the traditional Ghost image files were split into a 2GB file and a 1GB file. Based on the graphs in section 4.1, the average operation is 7MB larger while caching linked clones than imaging Ghost images. Thus, overall, it appears that throughput, and therefore restoration and archival time, depends on operational characteristics (read versus write) and operation size. The cache-and-update approach offers a far more efficient and faster approach to archiving and restoring user-specific models.

4.3   Workstation Capabilities

The sampled workstations had no issues executing three linked clones, with the specified virtual memory allocation. All linked clones as well as the host operating system were responsive. Neither XP nor Vista appeared to render a more or less responsive model. Based upon the capabilities of the sampled workstations, one could assume that other workstations with similar computing resources could execute the same number of workstations. Finally, the operating capacity of the workstations was not sought but rather a capability was determined; the workstations could likely handle executing more linked clones.

The workstations, with 3GB of memory and a 3.4 GHz processor, in the sampled environment were easily able to operate multiple (3) virtual machines simultaneously. It is believed that workstations with similar hardware will have similar capabilities. Since, in this environment, users control their virtual machines, the exercise of resource allocation between the workstation and the virtual machines it operates is a user responsibility. Therefore, users must learn to gauge virtual machine resource allocation based on total available resources. It appears that the recommended operating system specific virtual memory allocation by VMware in the Workstation product lends usable virtual environments. Users must pay attention to the aggregate memory of all operating virtual machines to ensure that the expected resource use does not exceed that provided by the workstation. Thus, in the researched environment, users get more operational benefits from individual workstations. This translates to the possibility of decreased workstation cost or increased lab utilization – both of which benefit the organization.

5 Conclusions

Since modeling is a pervasive scientific query and this research presents a usable and scalable modeling environment, this research is important. The problem of providing a manageable, usable, and scalable modeling environment can be solved by using an implementation of the researched environment. Organizations employing system engineers such as software developers, operating system programmers, system administrators, forensic analysts, malware researchers, students, and industry trainees could benefit from the new environment. The new environment satisfies basic modeling requirements by using a virtualization platform where users can model entire systems with benefits like model snapshots and modularity. The new environment is manageable, in the sample environment, because there are processes and methods to setup and maintain the infrastructure, deploy workstations, create templates and issue template refreshes. Further, through automation capabilities, the cost for management decreases as automation will reduce human-error and the time required for each task. It is clear that the new environment is a further scalable approach to system modeling than the traditional approaches based on the results from the restoration time experiment. Since multiple linked clones can operate per workstation, resource utilization of individual workstations is increased, each user requires fewer workstations, and, therefore, the overall modeling environment efficiency is increased. Since template-based virtual machines are used, the storage and network requirement per model is decreased which results in less time to preserve models and a higher rate of use of workstations. If an organization operates many powerful workstations, this modeling approach will decrease the time required for user model preservation and decrease hardware dependency while providing large-scale system modeling.

6 Future Work

In theory, if users were not required to cache their differential virtual machine prior to executing it, even less time is required for restoration and preservation. In future experiments, attention should be focused to understand the requirements for simultaneous direct modification of many linked clones. If the new environment were componentized into storage, network, and workstation items, one could posit that improving the capabilities of a component item might increase throughput during direct modification. Since more workstations means more models are capable of running, the number of workstations can cause poor throughput. However, increasing the capabilities of a workstation will not increase throughput. Therefore, it is reasonable to assume that the problem lies with either the network or the storage solution. Therefore, one could use multiple storage solutions or networks in large-scale direct modification experiments and observing which configuration yields the most usable models. Further, individual hardware components could be configured differently so they are tailored to linked clone behaviors. For example, since it is understood that the actual operation of a linked clone requires nearly equal read and write operations, a disk setup requiring less physical operations per logical operation might prove beneficial; that is to say, a RAID stripped and mirrored (RAID 0+1 or RAID 1+0) requires less physical disk operations per logical operation.

In addition to simply understanding direct modification requirements, vendors like VMware are constantly releasing new products that enable virtualization and virtualization management. For example, products like VMware ACE (VMware, Inc. 2008c) or Offline VDI (Lowe 2008) might enhance the ability to control virtual machines across many workstations. VMware ACE enables administrators to expire virtual machines at a specified date as well as prevent the virtual machine from operating outside of their organization. Recently introduced at VMworld 2008, the concept of Offline VDI enables users of VMware Virtual Desktop Infrastructure to "to check out [virtual machines] and run them while disconnected from the [VMware Virtual Infrastructure] environment" (Lowe 2008).

7 Appendices

7.1 Restoration Result Database

```
traditional,60,maul101,8935.3245675
traditional,60,maul102,8839.7470542
traditional,60,maul103,8659.2169596
traditional,60,maul104,8885.3405124
traditional,60,maul11,8969.2774752
traditional,60,maul12,8797.3410756
traditional,60,maul13,8689.7636391
traditional,60,maul14,8575.9674924
traditional,60,maul21,8722.3571805
traditional,60,maul22,8713.84161
traditional,60,maul24,8676.7637223
traditional,60,maul31,8891.9342202
traditional,60,maul32,8926.6371231
traditional,60,maul33,8876.9811909
traditional,60,maul34,8938.1682993
traditional,60,maul41,8881.4186625
traditional,60,maul42,8898.4810533
traditional,60,maul43,8946.6369951
traditional,60,maul44,9031.8708246
traditional,60,maul61,8971.3868367
traditional,60,maul63,8867.5437513
traditional,60,maul64,8819.4971838
traditional,60,maul71,8737.8727062
traditional,60,maul72,8883.7623975
traditional,60,maul73,8632.2483822
traditional,60,maul74,8553.7332597
traditional,60,maul81,8537.7177372
traditional,60,maul82,8480.5306032
traditional,60,maul83,8528.4052968
traditional,60,maul84,8903.8716438
traditional,60,maul92,8896.2466926
traditional,60,maul93,8700.6073197
traditional,60,maul94,8505.8898159
traditional,60,sidious102,8668.1387775
traditional,60,sidious103,8610.6860202
traditional,60,sidious13,8694.9511059
traditional,60,sidious21,8667.8262795
traditional,60,sidious22,8788.9661292
traditional,60,sidious23,8764.5756603
traditional,60,sidious33,8573.3893839
traditional,60,sidious34,8719.1853258
traditional,60,sidious41,8688.7323957
traditional,60,sidious42,8892.3717174
traditional,60,sidious43,8894.1685809
traditional,60,sidious51,8892.8560893
traditional,60,sidious52,8836.1689521
traditional,60,sidious53,8940.7932825
traditional,60,sidious61,8939.8557885
traditional,60,sidious63,8923.8871407
traditional,60,sidious64,8902.8716502
```

```
traditional,60,sidious72,8866.2625095
traditional,60,sidious74,8733.997731
traditional,60,sidious82,8754.8882223
traditional,60,sidious83,8820.403428
traditional,60,sidious84,8731.8883695
traditional,60,sidious92,8677.9512147
traditional,60,sidious93,8872.6218438
traditional,30,maul11,4566.377025
traditional,30,maul12,4450.8465144
traditional,30,maul13,4412.2061367
traditional,30,maul14,4202.8793514
traditional,30,maul21,3669.8671377
traditional,30,maul22,4331.1285306
traditional,30,maul24,4005.1774917
traditional,30,maul31,4150.1921886
traditional,30,maul32,3965.8964931
traditional,30,maul33,3953.1934494
traditional,30,maul34,4060.3490136
traditional,30,maul41,4423.6904382
traditional,30,maul42,4453.1433747
traditional,30,maul43,4157.4577671
traditional,30,sidious103,4480.2525762
traditional,30,sidious21,4164.4264725
traditional,30,sidious22,3900.8656593
traditional,30,sidious23,3883.6313946
traditional,30,sidious31,4004.7243696
traditional,30,sidious34,4278.722616
traditional,30,sidious41,4438.5028434
traditional,30,sidious42,3793.569471
traditional,30,sidious43,4370.2220304
traditional,30,sidious51,4445.4090492
traditional,30,sidious52,4250.4259221
traditional,30,sidious53,4389.2687835
traditional,30,sidious84,4478.4244629
traditional,30,sidious92,4510.8773802
traditional,30,sidious93,4502.9243061
traditional,1,maul11,1294.0385931
diffvirt-vista,70-17,maul101,479.5376677
diffvirt-vista,70-17,maul103,240.2610119
diffvirt-vista,70-17,maul11,242.8859615
diffvirt-vista,70-17,maul12,242.0891018
diffvirt-vista,70-17,maul13,149.4033814
diffvirt-vista,70-17,maul14,243.9796905
diffvirt-vista,70-17,maul21,155.6532614
diffvirt-vista,70-17,maul22,554.1612349
diffvirt-vista,70-17,maul24,369.2272857
diffvirt-vista,70-17,maul31,322.1656893
diffvirt-vista,70-17,maul32,353.3994646
diffvirt-vista,70-17,maul33,371.3834943
diffvirt-vista,70-17,maul34,416.3045068
diffvirt-vista,70-17,maul64,525.8024044
diffvirt-vista,70-17,maul71,584.5512764
diffvirt-vista,70-17,maul82,377.7739966
diffvirt-vista,70-17,maul94,535.2865973
diffvirt-vista,70-17,sidious101,522.8493361
diffvirt-vista,60,maul101,273.5416229
diffvirt-vista,60,maul103,272.9010102
```

```
diffvirt-vista,60,maul14,281.4945952
diffvirt-vista,60,maul21,247.4171245
diffvirt-vista,60,maul22,279.2602631
diffvirt-vista,60,maul31,273.3228771
diffvirt-vista,60,maul34,274.6978507
diffvirt-vista,60,maul41,276.9165581
diffvirt-vista,60,maul61,268.1979755
diffvirt-vista,60,maul63,271.7291577
diffvirt-vista,60,maul64,265.3542801
diffvirt-vista,60,maul71,264.4011734
diffvirt-vista,60,maul73,257.7606759
diffvirt-vista,60,maul81,259.6043905
diffvirt-vista,60,maul83,257.4169325
diffvirt-vista,60,maul92,255.0888522
diffvirt-vista,60,sidious104,343.587153
diffvirt-vista,60,sidious11,331.2905141
diffvirt-vista,60,sidious12,290.1975531
diffvirt-vista,60,sidious21,257.963797
diffvirt-vista,60,sidious22,256.6356975
diffvirt-vista,60,sidious23,253.12014
diffvirt-vista,60,sidious32,309.9159245
diffvirt-vista,60,sidious33,311.7283897
diffvirt-vista,60,sidious41,319.6032385
diffvirt-vista,60,sidious61,323.6500358
diffvirt-vista,60,sidious63,322.9312996
diffvirt-vista,60,sidious64,323.118796
diffvirt-vista,60,sidious74,292.5568828
diffvirt-vista,60,sidious81,336.2591687
diffvirt-vista,60,sidious84,325.7906197
diffvirt-vista,30,maul101,231.5424293
diffvirt-vista,30,maul13,187.3557777
diffvirt-vista,30,maul14,220.6832628
diffvirt-vista,30,maul21,186.2151746
diffvirt-vista,30,maul22,214.8240003
diffvirt-vista,30,maul24,234.6986187
diffvirt-vista,30,maul34,229.1362255
diffvirt-vista,30,maul41,222.0582364
diffvirt-vista,30,maul43,224.1206968
diffvirt-vista,30,maul64,226.0581596
diffvirt-vista,30,maul71,226.7612711
diffvirt-vista,30,maul72,226.9800169
diffvirt-vista,30,maul73,226.2769054
diffvirt-vista,30,maul81,232.9799017
diffvirt-vista,30,maul82,193.6994059
diffvirt-vista,30,maul84,239.0735347
diffvirt-vista,30,maul94,197.6837044
diffvirt-vista,30,sidious102,212.4021718
diffvirt-vista,30,sidious103,248.1983595
diffvirt-vista,30,sidious104,240.4328836
diffvirt-vista,1,maul11,288.5100855
diffvirt-xp,60,maul101,363.0479442
diffvirt-xp,60,maul102,244.1140632
diffvirt-xp,60,maul103,336.8767392
diffvirt-xp,60,maul104,350.2670214
diffvirt-xp,60,maul11,201.7253373
diffvirt-xp,60,maul13,192.9288564
diffvirt-xp,60,maul14,337.9860858
```

```
diffvirt-xp,60,maul21,256.3478901
diffvirt-xp,60,maul22,293.8465908
diffvirt-xp,60,maul31,359.0167974
diffvirt-xp,60,maul32,187.2416112
diffvirt-xp,60,maul34,349.1732994
diffvirt-xp,60,maul41,262.4413671
diffvirt-xp,60,maul42,241.7704182
diffvirt-xp,60,maul43,318.3615882
diffvirt-xp,60,maul61,267.4255188
diffvirt-xp,60,maul62,169.5392793
diffvirt-xp,60,maul63,351.1107498
diffvirt-xp,60,maul64,316.299141
diffvirt-xp,60,maul71,338.0485842
diffvirt-xp,60,maul72,343.4234466
diffvirt-xp,60,maul73,358.0324476
diffvirt-xp,60,maul74,342.0484818
diffvirt-xp,60,maul81,340.6110186
diffvirt-xp,60,maul82,284.6437014
diffvirt-xp,60,maul83,231.0521484
diffvirt-xp,60,maul84,356.9855994
diffvirt-xp,60,maul92,318.7365786
diffvirt-xp,60,maul93,359.3136648
diffvirt-xp,60,maul94,352.4857146
diffvirt-xp,60,sidious102,355.1418966
diffvirt-xp,60,sidious103,365.751
diffvirt-xp,60,sidious104,328.9863162
diffvirt-xp,60,sidious11,357.5637096
diffvirt-xp,60,sidious12,354.9856506
diffvirt-xp,60,sidious13,352.079475
diffvirt-xp,60,sidious14,350.517015
diffvirt-xp,60,sidious21,316.2522672
diffvirt-xp,60,sidious22,331.767495
diffvirt-xp,60,sidious23,335.673645
diffvirt-xp,60,sidious31,270.2378928
diffvirt-xp,60,sidious32,365.6885016
diffvirt-xp,60,sidious33,356.7356058
diffvirt-xp,60,sidious34,352.2044718
diffvirt-xp,60,sidious41,345.1421526
diffvirt-xp,60,sidious43,339.8297886
diffvirt-xp,60,sidious51,346.5171174
diffvirt-xp,60,sidious52,343.329699
diffvirt-xp,60,sidious53,345.6421398
diffvirt-xp,60,sidious54,332.9549646
diffvirt-xp,60,sidious61,356.4856122
diffvirt-xp,60,sidious63,365.8603722
diffvirt-xp,60,sidious64,345.3921462
diffvirt-xp,60,sidious72,329.267559
diffvirt-xp,60,sidious73,353.25132
diffvirt-xp,60,sidious74,343.564068
diffvirt-xp,60,sidious81,348.8139336
diffvirt-xp,60,sidious82,343.7671878
diffvirt-xp,60,sidious83,328.3144584
diffvirt-xp,60,sidious84,351.766983
diffvirt-xp,30,maul101,196.3043926
diffvirt-xp,30,maul102,175.7581751
diffvirt-xp,30,maul103,195.1325551
diffvirt-xp,30,maul104,195.3669226
```

```
diffvirt-xp,30,maul71,205.9603336
diffvirt-xp,30,maul72,207.2102936
diffvirt-xp,30,maul73,200.6792526
diffvirt-xp,30,maul74,215.8975156
diffvirt-xp,30,maul81,186.7421986
diffvirt-xp,30,maul82,191.6014181
diffvirt-xp,30,maul83,207.4915346
diffvirt-xp,30,maul84,216.3193771
diffvirt-xp,30,maul92,208.4602536
diffvirt-xp,30,maul93,208.9758621
diffvirt-xp,30,maul94,203.4604136
diffvirt-xp,30,sidious11,210.8508021
diffvirt-xp,30,sidious12,213.3819711
diffvirt-xp,30,sidious13,220.3348736
diffvirt-xp,30,sidious14,217.5380881
diffvirt-xp,30,sidious21,221.8504501
diffvirt-xp,30,sidious22,213.5538406
diffvirt-xp,30,sidious23,229.3502101
diffvirt-xp,30,sidious31,201.3979796
diffvirt-xp,30,sidious32,223.2566551
diffvirt-xp,30,sidious33,213.7725836
diffvirt-xp,30,sidious34,197.9605896
diffvirt-xp,30,sidious81,229.7251981
diffvirt-xp,30,sidious82,225.8034486
diffvirt-xp,30,sidious83,206.6321871
diffvirt-xp,30,sidious84,226.5690491
diffvirt-xp,1,maul102,190.1838484
```

7.2 diffVirtResult.ps1

```
# diffVirtResult.ps1
# written in PowerShell 1.0
# script maps a network share based on its IP address and writes a file to
the share
#
#
#this script is
#-set to execute on startup
#-used for automating a differential virtualization restoration experiment
#-written in PowerShell
#-written by Jason Koppe
#

#identify imaging subnets
# 10.200.251.0/24 is maul
# 10.200.250.0/24 is sidious
$names = @{"251" = "maul"; "250" = "sidious"}

#select and split the output of ipconfig to get the imaging NIC ip address
$ip=(ipconfig | select-string "10.200.25" | select-string
"IP").tostring().split(":")[1]
```

```
#third octet of the IP address is side-specific
$subnet=$ip.split(".")[2]

#fourth octet of the IP address is machine-specific
$node=$ip.split(".")[3]

#quick lookup to the hash
$name=$names[$subnet].tostring()
$name+=$node

#copy the linked clone
del c:\users\$name\desktop\* -recurse -force
robocopy z:\restoration c:\users\$name\desktop\

#start the linked clone and start a script in the linked
#clone based on the host name
& 'C:\Program Files\VMware\VMware Workstation\vmrun.exe' start
"c:\users\$name\desktop\Clone of Restoration.vmx"
& 'C:\Program Files\VMware\VMware Workstation\vmrun.exe' -gu administrator -
gp netsys runprograminguest "c:\users\$name\desktop\Clone of Restoration.vmx"
c:\windows\system32\windowspowershell\v1.0\powershell.exe "c:\koppe.ps1
$name"
```

7.3   batchanalyze.ps1

```
#batchanalyze.ps1
#written in Windows PowerShell 1.0
#
#Given an input file of format:
#     testname,testsize,workstationname,restorationtime
#Output analysis statistics for all test types at each heat size

#data analysis types
$types = "avg","min","max"

#set default analysis type to average
$type = $types[0]

#set default database file name
$db = "s.csv"

#if there are command line arguments, use them to set db and type
if ($args.count -ge 1 ) {
     $db = $args[0]
}
if ($args.count -eq 2 ) {
     $type = $args[1]
}


#test names
$tests = "traditional","diffvirt-xp","diffvirt-vista"

#test sizes
```

```
$sizes = 1,30,60


#for each test type
for($t=0;$t -lt $tests.length;$t++){
      $testline = ""

      #for each test size
      for($s=0;$s -lt $sizes.length;$s++) {


            $val = 0
            $min = 99999999999999999999
            $max = 0
            $sum = 0
            $count = 0
            $name=$tests[$t]
            $size=$sizes[$s]

            #for each line in the database, split on comma and update sum,
min, max, count
            gc $db | foreach {
                  $line = $_.split(",")
                  if (($line[0] -eq $name) -and ($line[1] -eq $size)) {
                        $sum+=$line[3]
                        if ($line[3] -gt $max) { $max = $line[3] }
                        if ($line[3] -lt $min) { $min = $line[3] }
                        $count++
                  }
            }
            #calculate average
            if ($count -eq 0) { $avg = 0 } else { $avg = $sum/$count }

            #expectations of count met?
            if ($count -ne $size) { "ERROR: Number of records ($count)
doesn't match inputted test size ($size)" }
            switch ($type) {
                  'avg' { $testline+=[math]::round($avg,3) }
                  'min' { $testline+=[math]::round($min,3) }
                  'max' { $testline+=[math]::round($max,3)   }
            }
            if ($s -ne ($sizes.length-1)) { $testline+="," }

      }

      #write output
      Write-Host $testline
}
```

7.4   starttest.ps1

```
#restoration test start script
#Written in PowerShell 1.0
#the purpose of this script is to record the time on the server at the
```

```
#start of a test and generate a record for each workstation at the end of a
#test

#get testname and size
$name = Read-Host "Enter test name and size (Ex: Traditional,30)"

#Get the date
$date = get-date

#Pause until the end of the test
Read-Host "Begin the test now & hit enter when the test is done..."

#for each folder in the directory, find the result file
#for each result file, calculate the difference between its creation time
#and the start of the test and record this in the database.
#finally rename the result file for longevity
$basefolder = get-item "e:\students\20074\599-01\"
Get-ChildItem $basefolder  | foreach-object {
      $path = "$basefolder\$_\result"
      if ((Test-Path -Path $path)  -eq $True) {
            $result = get-childitem $path
            $diff = $result.creationTime.subtract($date).totalseconds
            $record = "$name,$_,$diff"
            $record >> restoration.csv
            Rename-Item -Path $path -newname $record
      }
}
```

7.5 massImage.ps1

```
# massImage.ps1
# written in Powershell CTP 2.0
# The purpose of this script is to automate configuration of a multicast
# transmission and unattended installation of a specific install image on a
# specific WDS server.  A detailed process overview is given below.
########
#Prerequisites
#
#-Windows 2008
#-PowerShell CTP 2.0
#-WDSUTIL
#-Administrative shell
#-Domain credentials

########
#Process
#
#-select wds server
#-select image
#-generate client unattend
#-unique client unattend name
#-store client unattend
#-store boot program
#-set client unattend
```

```
#-enable n12 as boot program
#-check pre-existing MC sessions
#-create multicast session
#-pause until trigger from input
#-check # of clients connected
#-start multicast
#-revert boot program
#-revert client unattend

########
#Author
#-Jason R Koppe

##########
#Global Variables
$wds
$images
$xmls
$running = "RUNNING"
$stopped = "STOPPED"
$server
$wdsoutput

##########
#Functions
Function Prompt-YesNo ($Caption, $Message,$choices) {

$host.ui.PromptForChoice($caption,$Message,[System.Management.Automation.Host
.ChoiceDescription[]]$choices,0)
}

Function wdsState($s) {
      $state = (sc.exe \\$s query wdsserver | Select-String "STATE")
      if ($state) { $state = $state.tostring().split(":")[1].trim().split("
")[2] }
      if ($state -eq $running) {
            return $running
      }
      return $stopped
}

#########
#Check for prerequisites
Write-Host "Prerequisites"
Write-Host "-------------"
Write-Host "Checking Windows version...`t" -NoNewline

if ((get-wmiobject -class "Win32_OperatingSystem" -namespace "root\CIMV2"  -
computername .).name -match "2008") {
      Write-Host "ok" -ForegroundColor "green"
} else {
      Write-Host "failed" -ForegroundColor "red"
      Write-Host "Please run this script on Windows Server 2008" -
ForegroundColor "red"
      exit
}
```

```powershell
Write-Host "Checking PowerShell version...`t" -NoNewline

if ((get-pssnapin -name Microsoft.Powershell.core).psversion.major -ge 2) {
      Write-Host "ok" -ForegroundColor "green"
} else {
      Write-Host "failed" -ForegroundColor "red"
      Write-Host "Please install PowerShell 2.0 (Select-string -context is
required)" -ForegroundColor "red"
      exit
}

Write-Host "Checking privileges...`t`t" -NoNewline
servermanagercmd > $null 2>$null
if ($lastexitcode -eq 4) {
      Write-Host "ok" -ForegroundColor "green"
} else {
      Write-Host "failed" -ForegroundColor "red"
      Write-Host "Please run this from an Administrative console" -
ForegroundColor "red"
      exit
}

Write-Host "Checking WDS tools...`t`t" -NoNewline

if ((Test-Path C:\Windows\System32\wdsutil.exe) -eq $true) {
      Write-Host "ok" -ForegroundColor "green"
} else {
      Write-Host "failed" -ForegroundColor "red"

      $n = ([System.Management.Automation.Host.ChoiceDescription]"&No")
      $n.helpmessage = "No, don't install WDS tools"

      $Y = ([System.Management.Automation.Host.ChoiceDescription]"&Yes")
      $y.helpmessage = "Yes, install netsh WDS tools"
      $choices = ($Y,$N)


      $ans = $host.ui.PromptForChoice("Install WDS tools","Would you like to
install WDS utilities now?",

      [System.Management.Automation.Host.ChoiceDescription[]]$choices,0)
      if ($ans -eq 0) {
            servermanagercmd -install RSAT-WDS
      }else { exit }
}

############
#Main script
Write-Host "`n`nMulticast Imaging"
Write-Host "-----------------"

if ($args.count -eq 1) {
      Write-Host "Validating input server...`t" -NoNewline
      $s = $args[0]
      wdsutil /get-allimages /server:$s /show:Install /detailed > $null
      if ($lastexitcode -eq 0) {
            $server = $s
```

```powershell
                Write-Host "ok" -ForegroundColor "green"
        }
        else {
                Write-Host "failed"  -ForegroundColor "red"
        }
}
if (-not $server) {
        Write-Host "Finding WDS servers...`t`t" -NoNewline
        $wdsoutput = wdsutil /get-allservers /show:config

        $wdsoutput | select-string "Attempting to contact server" | foreach {
                $wds += ,($_.tostring().split(" ")[4])
        }
        if ($wds.count -gt 0) {
                Write-Host "done" -ForegroundColor "green"
        } else {
                Write-Host "done" -ForegroundColor "red"
                Write-Host "No WDS servers found" -ForegroundColor "red"
                if (${env:userdomain} -eq ${env:computername}) {
                        Write-Host "Run this console from a domain admin account" -
ForegroundColor "red"
                }
                Write-Host "Try running the script with the servername as the
parameter" -ForegroundColor "red"
                Write-Host "`tEx: C:\admin\scripts\massImage.ps1 srv1" -
ForegroundColor "red"
                exit
        }


        Write-Host "Checking WDS state...`t`t" -NoNewline
        #add .state to string objects in the wds collection
        for ($i = 0; $i -lt $wds.count; $i++) {
                $state = wdsState $wds[$i]
                $wds[$i] = $wds[$i] | add-member noteproperty state $state -
passthru
        }
        Write-Host "done`n" -ForegroundColor "green"

        #get choice server
        $lc = 0
        while($server.state -ne $running) {

                #print menu on first iteration
                if ($lc -eq 0) {
                        write-host "[#] Server"
                        write-host "----------"
                        for ($i = 0; $i -lt $wds.count; $i++) {
                                $w = $wds[$i].tostring().split(".")[0]
                                $s = $wds[$i].state

                                if ($s -eq $running) { write-host [$i] $w -
ForegroundColor "green" }
                                else { write-host [$i] $w }
                        }
                }
```

```
            #print warning after first iteration
            if ($lc -gt 0) { Write-Host "`nNOTE: WDS must be running on the
server.`n" }

            #get input
            $in = read-host "Select a WDS server"

            #validate input
            if (($in -cmatch "^\d+$") -eq $true) {
                    $in = [int]$in
                    if (($in -ge 0) -and ($in -lt $wds.count)) {
                            $server = $wds[$in]
                    }
            }

            $lc++
      }
}


#find images on $server
Write-Host "Finding images...`t`t" -NoNewline
$wdsoutput = wdsutil /get-allimages /server:$server /show:Install /detailed

$wdsoutput | select-string "Image name" -context 0,4| foreach-object {

      $name = $_.line.tostring().split(":")[1].trim()
      $group = ($_.context.postcontext | select-string
"group").tostring().split(":")[1].trim()

      $image = $name
      $image = $image | Add-Member noteproperty group $group -PassThru
      $image = $image | Add-Member noteproperty server $server -PassThru
      $images += ,$image
}
Write-Host "ok`n" -ForegroundColor "green"

#get choice image
$simage
$simagegroup
$lc = 0
while(-not $simage) {

      #print menu on first iteration
      if ($lc -eq 0) {
            write-host "[#] Group | Name "
            write-host "-------------------"
            for ($i = 0; $i -lt $images.count; $i++) {
                    $n = $images[$i].tostring()
                    $g = $images[$i].group
                    write-host "[$i] $g | $n"
            }
      }

      #print warning after first iteration
      if ($lc -gt 0) { Write-Host "`nNOTE: Input a valid number" }
```

```
        #get input
        $in = read-host "`nSelect an image"

        #validate input
        if (($in -cmatch "^\d+$") -eq $true) {
                $in = [int]$in
                if (($in -ge 0) -and ($in -lt $images.count)) {
                        $simage = $images[$in]
                        $simagegroup = $images[$in].group
                }
        }

        $lc++
}
Write-Host "Generating unattend name...`t" -NoNewline

#find full path for WdsClientUnattend
$reminst = (wdsutil /get-server /show:all /detailed | select-string "REMINST
location").tostring().trim().split(" ")[2]

#unique name for temporary unattend xml
$xfile = "" + (new-object random).next() + ".xml"
$fullname = "$reminst\WdsClientUnattend\$xfile"

#verify unique name for temporary unattend xml
while ((test-path $fullname) -eq $true) {
        $xfile = "" + (new-object random).next() + ".xml"
        $fullname = "$reminst\WdsClientUnattend\$xfile"
}
Write-Host "ok" -ForegroundColor "Green"
Write-Host "Generating unattend file...`t" -NoNewline
#generate temporary unattend xml
set-content $fullname (get-content default.xml | foreach {
        $ng = ">$simagegroup<"
        $_ -replace ">IMAGEGROUP<", $ng } | foreach {
                $_ -replace ">IMAGENAME<", ">$simage<"
})
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }

#save boot program
$orig_bootprogram = (wdsutil /get-server /show:all /detailed | select-string
"default boot programs:" -context 0,1 | foreach-object {
        ($_.context.postcontext | select-string
"x86").tostring().trim().split(" ")[3]
})
#save boot program 64
$orig_bootprogram64 = (wdsutil /get-server /show:all /detailed | select-
string "default boot programs:" -context 0,2 | foreach-object {
        ($_.context.postcontext | select-string
"x64").tostring().trim().split(" ")[3]
})
#save client unattend
$orig_clientunattend = (wdsutil /get-server /show:all /detailed | select-
string "WDS unattend files:" -context 0,1 | foreach-object {
        ($_.context.postcontext | select-string
```

```powershell
"x86").tostring().trim().split(" ")[3]
})


#set new boot program
Write-Host "Setting boot program...`t`t" -NoNewline
wdsutil /set-server /server:$server /bootprogram:boot\x86\pxeboot.n12
/architecture:x86  > $null
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }
#set new boot program
Write-Host "Setting boot program...`t`t" -NoNewline
wdsutil /set-server /server:$server /bootprogram:boot\x64\pxeboot.n12
/architecture:x64  > $null
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }
Write-Host "Setting unattend file...`t" -NoNewline
#set new unattend file
wdsutil /set-server /server:$server /wdsunattend
/file:WdsClientUnattend\WdsClientUnattend\$xfile /architecture:x86  > $null
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }
Write-Host "Checking multicast session...`t" -NoNewline
WDSUTIL /get-MulticastTransmission /Server:$server /Image:$simage
/ImageType:Install /imagegroup:$simagegroup > $null
if ($lastexitcode -eq -1056767648) {
      Write-Host "ok" -ForegroundColor "green"
      #create multicast session
      Write-Host "Creating multicast session...`t" -NoNewline
      WDSUTIL /New-MulticastTransmission /FriendlyName:"WDS SchedCast
Transmission" /Server:$server /Image:$simage /ImageType:Install
/imagegroup:$simagegroup /TransmissionType:ScheduledCast  > $null
      if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" }
else { Write-Host "failed" -ForegroundColor "red" }
}else {
      Write-Host "already existed"
      Write-Host "Deleting session...`t`t" -NoNewline
      #delete multicast session
      WDSUTIL /remove-MulticastTransmission /Server:$server /Image:$simage
/ImageType:Install /imagegroup:$simagegroup /force  > $null
      if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" }
else { Write-Host "failed" -ForegroundColor "red" }
      Write-Host "Recreating session...`t`t" -NoNewline
      #recreate multicast session
      WDSUTIL /New-MulticastTransmission /FriendlyName:"WDS SchedCast
Transmission" /Server:$server /Image:$simage /ImageType:Install
/imagegroup:$simagegroup /TransmissionType:ScheduledCast  > $null
      if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" }
else { Write-Host "failed" -ForegroundColor "red" }
}


#start multicast session
Write-Host "Wait for all clients to join, then press ENTER" -ForegroundColor
"red"
Read-Host

while ((wdsutil /get-multicasttransmission /Server:$server /Image:$simage
/imagetype:install /imagegroup:$simagegroup /show:clients | select-string
```

```powershell
"Clients Connected").line.split(":")[1].trim() -eq 0) {
      Write-Host "No clients have joined the session" -ForegroundColor "red"
      Write-Host "Wait for all clients to join, then press ENTER" -
ForegroundColor "red"
      Read-Host
}
WDSUTIL /start-MulticastTransmission /Server:$server /Image:$simage
/ImageType:Install /imagegroup:$simagegroup > $null

Write-Host "The clients should now be imaging`n" -ForegroundColor "red"
#########
#Cleanup
Write-Host "Performing cleanup"
Write-Host "------------------"
#tombstone multicast session (don't allow more to join & delete when all
done)
Write-Host "Mark session for deletion...`t" -NoNewline
WDSUTIL /remove-MulticastTransmission /Server:$server /Image:$simage
/ImageType:Install /imagegroup:$simagegroup > $null
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }

#reset boot program
Write-Host "Reset boot program...`t`t" -NoNewline
wdsutil /set-server /server:$server /bootprogram:$orig_bootprogram
/architecture:x86 > $null
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }

#reset boot program
Write-Host "Reset boot program...`t`t" -NoNewline
wdsutil /set-server /server:$server /bootprogram:$orig_bootprogram64
/architecture:x64 > $null
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }

#reset unattend file
Write-Host "Reset unattend...`t`t" -NoNewline
wdsutil /set-server /server:$server /wdsunattend /file:$orig_clientunattend
/architecture:x86 > $null
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }

#remove temporary file
Write-Host "Remove unattend file...`t`t" -NoNewline
Del $fullname -Force > $null
if ($lastexitcode -eq 0) { Write-Host "ok" -ForegroundColor "green" } else {
Write-Host "failed" -ForegroundColor "red" }
```

7.6 default.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
    <settings pass="windowsPE">
```

```xml
<component name="Microsoft-Windows-Setup"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
processorArchitecture="x86">
        <DiskConfiguration>
        <WillShowUI>OnError</WillShowUI>
            <Disk>
                <CreatePartitions>
                    <CreatePartition>
                        <Order>2</Order>
                        <Type>Primary</Type>
                        <Extend>true</Extend>
                    </CreatePartition>
                    <CreatePartition>
                        <Order>1</Order>
                        <Type>Primary</Type>
                        <Size>20000</Size>
                    </CreatePartition>
                </CreatePartitions>
                <ModifyPartitions>
                    <ModifyPartition>
                        <Active>true</Active>
                        <Format>NTFS</Format>
                        <Label>Public</Label>
                        <Letter>P</Letter>
                        <Order>2</Order>
                        <PartitionID>1</PartitionID>
                    </ModifyPartition>
                    <ModifyPartition>
                        <Active>true</Active>
                        <Extend>False</Extend>
                        <Format>NTFS</Format>
                        <Label>Local Disk</Label>
                        <Letter>C</Letter>
                        <Order>1</Order>
                        <PartitionID>2</PartitionID>
                    </ModifyPartition>
                </ModifyPartitions>
                <DiskID>0</DiskID>
                <WillWipeDisk>true</WillWipeDisk>
            </Disk>
        </DiskConfiguration>
        <ImageInstall>
            <OSImage>
                <InstallTo>
                    <DiskID>0</DiskID>
                    <PartitionID>2</PartitionID>
                </InstallTo>
                <WillShowUI>OnError</WillShowUI>
            </OSImage>
        </ImageInstall>
        <UserData>
            <ProductKey>
                <WillShowUI>OnError</WillShowUI>
            </ProductKey>
            <AcceptEula>true</AcceptEula>
            <FullName>IT</FullName>
            <Organization>RIT</Organization>
```

```
                    </UserData>
                    <WindowsDeploymentServices>
                        <Login>
                            <Credentials>
                                <Domain>TESTDOMAIN</Domain>
                                <Password>TESTPASSWORD</Password>
                                <Username>TESTUSERNAME</Username>
                            </Credentials>
                        </Login>
                        <ImageSelection>
                            <InstallImage>
                                <ImageGroup>IMAGEGROUP</ImageGroup>
                                <ImageName>IMAGENAME</ImageName>
                            </InstallImage>
                            <InstallTo>
                                <DiskID>0</DiskID>
                                <PartitionID>2</PartitionID>
                            </InstallTo>
                        </ImageSelection>
                    </WindowsDeploymentServices>
            </component>
            <component name="Microsoft-Windows-International-Core-WinPE"
processorArchitecture="x86" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS"
xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <SetupUILanguage>
                    <UILanguage>en-US</UILanguage>
                </SetupUILanguage>
                <InputLocale>en-US</InputLocale>
                <SystemLocale>en-US</SystemLocale>
                <UILanguage>en-US</UILanguage>
                <UserLocale>en-US</UserLocale>
            </component>
        </settings>
        <cpi:offlineImage cpi:source="" xmlns:cpi="urn:schemas-microsoft-com:cpi"
/>
</unattend>
```

7.7   useradd.pl

```
#!/usr/bin/perl -w
#Author: Jason Koppe
#Started 11/27/05
#Code Finished 3/12/06
#Comments Finished 3/15/06
#USAGE: perl <path to perl script> <path to ini file>
#Example: perl d:\admin\useradd.pl d:\admin\input\useradd.ini

#Items to add
########
#AD Groups like 071-421-39
```

```perl
#Class choices like 421-39
#Class path like \\jabba\students\20071\
#Automatic Date




use Term::ReadKey;
use Config::INI::Simple;

#3/10/06 JK
#Catches CTRL+C
$SIG{INT} = sub { print "\nProgram terminated by CTRL+C!!!\n\n"; exit 5; };

#11/27/05
#Sub taken from dhcp_config.pl written by Suraaj Gaur
#3/10/06 JK Modified
#     To check for undefined and reask the question
#     otherwise return chomped input
sub prompt {
      my ($message) = @_;
      print $message;
      my $input = <STDIN>;
      if (!(defined($input))) { print "\n"; prompt($message); }
      chomp($input);
      return $input;
}

#11/27/05
#Sub taken from dhcp_config.pl written by Suraaj Gaur
#Prompts for password, or other sensitive input.  Does not echo
#out text typed in
#3/10/06 JK Modified
#     Not defined, don't chomp
sub passPrompt {
      while (1 eq 1) {
            Term::ReadKey::ReadMode('noecho');
            print "Enter password: ";
            my $input = <STDIN>;
            print "\n";
            print "Enter password again: ";
            my $secinput = <STDIN>;
            if (defined($input)) { chomp($input); }
            if (defined($secinput)) { chomp($secinput); }
            Term::ReadKey::ReadMode(0);
            print "\n";
            if ($input eq $secinput) {
                  print "Passwords match\n";
                  return $input;
            }
            else {
                  print "Passwords do not match, try again\n";
            }
      }
}

#11/27/05 JK
```

```perl
#0 - User is enabled
#1 - User is disabled
sub userDisabled {
      if ( "@_" eq "" ) { return 1; }
      my ( $user ) = @_;
      if ( `dsquery user -samid \"$user\" -disabled -o samid` =~ /^$/ ) {
return 0; }
      else { return 1; }
}
#11/28/05 JK
#0 - User is not created
#1 - User is created
sub userCreated {
      if ( "@_" eq "" ) { return 1; }
      my ( $user ) = @_;
      if ( `dsquery user -samid \"$user\" -o samid 2>&1` =~ /^$/ ) { return
0; }
      else { return 1; }
}


#3/10/06 JK
#Trims leading and trailing whitespace
sub trim {
      my ($string) = @_;
      $string =~ s/^\s+//;
      $string =~ s/\s+$//;
      return $string;
}


#2/11/06 JK
#0 - Group invalid
#1 - Group valid
#The array @courses is populated earlier from the .ini settings file
sub validCourse {
      my ( $course, @courses ) = @_;
      foreach(@courses) {
            if ($course eq $_) { return 1; }
      }
      return 0;
}


#3/10/06 JK
#0 - Group doesn't exist
#1 - Group exists
#Checks to see whether the group is created in active directory
sub groupCreated {
      if ( "@_" eq "" ) { return 0; }
      my ( $group ) = @_;
      if ( `dsquery group $group 2>&1` =~ /failed/ ) { return 0; }
      else { return 1; }
}


#2/11/06 JK
#updated 3/5/06
#0 - Server doesn't exist
#1 - Server exists
#The @servers arrary is populated earlier from the .ini settings file
```

```perl
#The input can match any part of the server string from the ini file
sub validServer {
      my ( $server, @servers ) = @_;
      $server = lc($server);
      my ( $s, $i ) = 0;
      foreach(@servers) {
             if ($server eq $_) { return $_; }
             if (substr($server,0,1) eq substr($_,0,1)) { return $_; }
      }
      return 0;
}

#2/11/06 JK
#updated 3/5/06
#0 - Day invalid
#1 - Day valid
#Weekdays for now
sub validDay {
      my ( $day ) = @_;
      $day = lc($day);
      if ( $day eq "m" || $day eq "t" || $day eq "w" || $day eq "r" || $day
eq "f" ) { return 1; }
      else { return 0; }
}

#3/5/06 JK
#1 - Add was successful
#0 - Add failed
#Creates a group in active directory
sub createGroup {
      my ($group) = @_;
      if ( `dsadd group $group` =~ /succeeded/ ) { return 1; }
      else { return 0; }
}

#3/9/06 JK
#Assuming the description of the user is the path where they save files,
#This sub will print the location of a user as well as the directory
#where they save files.  The description has been used in the past to store
#the users path, this could be modified to use the homedirectory of the users
#object.  The two values are returned in the array
sub printLocDesc {
      if ( "@_" eq "" ) { return; }
      my ($user) = @_;
      my ($dn) = `dsquery user -samid \"$user\"`;
      my (@result) = split(/\n/,`dsget user $dn -desc`);
      print "\tObject Location:  $dn";
      print "\tSave Directory:\t$result[1]\n";
      return ($dn,$result[1]);
}

# Add course to ini
# JK
sub addCourse {
      if ( @_ ne 2 ) {
             return -1;
      }
```

```perl
}

#JK
#This sub reads settings from the ini file an also gathers user input.  The
info
#collected from the sub include the quarter, course, server and day.  If the
ini
#file is provided as the first input parameter, the script will use that
path, otherwise
#it defaults to \\netsys.labs\share\teams\Scripts\Input\useradd.ini because
thats where most of our scripts will
#be located.
sub getSettings {
      my ($server,$quarter,$qtrprompt,$course, $courses, $servers) = "";
      my ($inifile) = new Config::INI::Simple;
      my ($hostname) = `hostname`;
      if ($#ARGV+1 eq 1) { $inifile->read("$ARGV[0]"); }
      else { $inifile-
>read("\\\\netsys.labs\\share\\teams\\scripts\\useradd.ini"); }
      $quarter = "$inifile->{default}->{quarter}";
      if ($hostname =~ /vader/i) {
            $courses = "$inifile->{default}->{syslab}";
            $servers = "$inifile->{default}->{vader}";
      }
      elsif ($hostname =~ /homer/i) {
            $courses = "$inifile->{default}->{projects}";
            $servers = "$inifile->{default}->{homer}";
      }
      elsif ($hostname =~ /wizard/i) {
            $courses = "$inifile->{default}->{netlab}";
            $servers = "$inifile->{default}->{wizard}";
      }
      elsif ($hostname =~ /milton/i) {
            $courses = "$inifile->{default}->{voip}";
            $servers = "$inifile->{default}->{milton}";
      }
      @coursesary = split(",",$courses);
      @serversary = split(",",$servers);
      $course = "";
      $server = 0;
      while ( validCourse($course,@coursesary) eq 0 ) {
            $course = prompt("Please input the course number ($courses): ");
      }

      while ( $server eq 0 ) {
            $server = validServer(lc(prompt("Please input the storage server
($servers): ")),@serversary);
      }
      $coursedn = "\"CN=$quarter-
$course,OU=Groups,OU=Students,DC=netsys,DC=labs\"";
      if (groupCreated($coursedn) eq 0 )  { createGroup($coursedn); }
      return ($course, $server, $quarter);
}

#3/12/06 JK
```

```
#1 - Valid name
#2 - Invalid name
#Checks to make sure that the input only contains letters, spaces and dashes
sub validName {
      my ( $name ) = @_;
      if (!defined($name)) { return 0; }
      if ( $name =~ /^[a-zA-Z\-\s]+$/ ) { return 1; }
      else { return 0; }
}


#JK
#Input lots of information that was collected through the getSettings sub.
#Prompt for a username, first, last password and attempt to create the user.
#The path and the distinguished name of the user will be created after
#the username, first and last names are inputted.
#The username and path are returned.
sub userAdd {
      my ( $course, $coursedn, $server, $quarter, $user, $badinput ) = @_;
      my ( $dn, $pass, $path, $output, $first, $last ) = "";
      if ( $user eq "" ) { $user = prompt("Please input a username: "); }
      while (userCreated($user) == 1) {
            print "ERROR: The user $user already exists\n";
            printLocDesc($user);
            print "Please remove the FOLDER and ACCOUNT manually if that is
the desired name\n";
            $user = prompt("Please input another username: ");
      }
      if (!defined($first)) { $first = trim(prompt("Input first name: ")); }
      while (validName($first) eq 0) {
            print "ERROR: Invalid characters found.  Only letters, spaces and
dashes are allowed\n";
            $first = trim(prompt("Please try again: "));
      }
      if (!defined($last)) { $last = trim(prompt("Input last name: ")); }
      while (validName($last) eq 0) {
            print "ERROR: Invalid characters found.  Only letters, spaces and
dashes are allowed\n";
            $last = trim(prompt("Please try again: "));
      }
      while (groupCreated($coursedn) eq 0) {
            print "ERROR: $coursedn not created.\nManually create this and
add the user.\n";
      }
      if (groupCreated($coursedn) eq 0 )  { createGroup($coursedn); }

      $path = "\\\\$server\\students\\$quarter\\$course\\$user";
      $dn = "\"CN=$user,OU=Users,OU=Students,DC=netsys,DC=labs\"";

      #This loop will continue to run if the password isnt complex
      #It will return blank values and a 1 if the username is invalid, the
entire function will be run again
      #If everything worked, it returns the user name, path and a 0 so that
the program will continue
      while ($badinput ne 0) {
            $pass = passPrompt();
```

```perl
################################################################
#$output = DSOut(`dsadd user $dn -pwd \"$pass\" -fn \"$first\" -ln \"$last\"
-display \"$first $last\" -desc \"$path\" -memberof
\"CN=Students,OU=Groups,OU=Students,DC=netsys,DC=labs\" $coursedn 2>&1`);
################################################################
#############Alex Modified Here, Don't beat me#################
################################################################
            $homedrv = "\\\\\jabba\\students\\$quarter\\$course\\$user";
            $output = DSOut(`dsadd user $dn -pwd \"$pass\" -fn \"$first\" -ln
\"$last\" -display \"$first $last\" -desc \"$path\" -hmdir \"$homedrv\" -
hmdrv \"S:\" -memberof
\"CN=Students,OU=Groups,OU=Students,DC=netsys,DC=labs\" $coursedn 2>&1`);
################################################################
################################################################

            if ($output =~ /complexity/) {
                    system("dsrm $dn -noprompt > NUL 2>&1");
            }
            elsif ($output =~ /not a properly formed account name/) {
                    return ("","",1);
            }
            elsif ($output =~ /succeeded/) {

                    return($user,$path,0);
            }
        }

}

#This will analyze the output of dsadd, dsquery, and dsget type programs.
#If the command fails, the error will be returned.  Otherwise, the succeeded
#message will be returned.
sub DSOut {
    chomp(my ( $output ) =  @_);
      my ( @result ) = split(/:/,$output);
      if ( $result[0] =~ /failed/ ) {
            if (defined($result[2])) { print "ERROR: $result[2]\n"; return
$result[2]; }
            else { print "ERROR: $result[1]\n"; return $result[1]; }
      }
      return $result[0];
}




##THE SCRIPT
##START CALLING ALL THE FUNCTIONS
my ( $user, $dn, $path, $quarter, $course, $server,  $coursedn, $section,
$sectiondn, $again ) = "";
my $keep = 0;#0 = Don't keep/1=keep settings
my $adduser = 1;#1 - Keep adding a user (and get settings if necessary)
my $badinput = 1;#1 - Bad input, to re run the useradd function

while ( $adduser eq 1 ) {
```

```perl
      if ($keep eq 0) {
              ($course, $server, $quarter) = getSettings();
              $coursedn = "\"CN=$quarter-
$course,OU=Groups,OU=Students,DC=netsys,DC=labs\"";
              print "Example Student Path:
\\\\$server\\$quarter\\$course\\abc1234\\\n";
              if (prompt("Keep these settings for all users created from now
on? (y/n) [y]: ") =~ /^n/i ) {
                      $keep = 0;
              }
              else {
                      $keep = 1;
                      print "\nSETTINGS SAVED!\nQuarter: $quarter\nServer:
$server\nCourse: $course\n\n";
              }
      }



      while ($badinput ne 0) {
              ($user,$path,$badinput) = userAdd($course, $coursedn, $server,
$quarter, $user, $badinput);
              $dn = "\"CN=$user,OU=Users,OU=Students,DC=netsys,DC=labs\"";
      }

      if ( userDisabled($user) == 0 && userCreated($user) == 1 ) {
              #Check if the path exists, if not, try to create it
              #$pathexist values
              #0 - Made successfully
              #1 - Already existed
              #Any other number - Unable to make the path
              if ( -d "\"$path\"" ) {
                      $pathexist = 1;
                      print "Path existed; the NTFS permissions for the folder
should be examined below.\n";
                      system("cacls \"$path\"");
              }
              else { $pathexist = system("mkdir \"$path\""); }

              #Make sure it was created successfully & fix NTFS permissions
              if ( $pathexist == 0 ) {
                      print "$path created successfully\n";
                      $caclsuser = system("cacls \"$path\" \/E \/G
\"NETSYS\\$user\":C > NUL");
                      $caclsgroup = system("cacls \"$path\" \/E \/R \"Users\" >
NUL");
                      if ( $caclsuser == 0 ) { print "Change control to \"$path\"
granted to $user\n"; }
                      else { print "There was an error giving the user full
access to the users directory\nPlease contact a NetSys Lab Server
Administrator.\n"; }
                      if ( $caclsgroup == 0 ) { print "Users permissions revoked
for \"$path\"\n"; }
                      else { print "There was an error removing the Users Read
permissions on the users directory\nPlease contact a NetSys Lab Server
Administrator.\n"; }
```

```
            }
            elsif ( ! -d $path && $pathexist != 0) {
                    print "WARNING: The path could not be created.  Program
terminating\nContact a NetSys Lab Server Administrator.\n";
                    exit 1;
            }
      }

      #Prompt to see if the script will loop again
      $again = "f";
      while (!($again =~ /^[yn]/i)) {
            $again = prompt("Would you like to add another user? (y/n): ");
            if ($again =~ /^y/i) {
                    $badinput = 1;
                    $adduser = 1;
                    $user = "";
            }
            elsif ($again =~ /^n/i) {
                    $adduser = 0;
                    $user = "";
            }
      }
}
```

7.8 startnet.cmd

```
@echo off
wpeinit
rundll32.exe setupapi,InstallHinfSection DefaultInstall 132 wimfltr.inf
wdsmcast /transfer-file /server:192.168.66.140 /namespace:"wimtest"
/username:koppe\joe /password:asdf1234! /sourcefile:temp.wim
/destinationfile:c:\temp.wim
mkdir c:\mount
imagex /mount c:\temp.wim 1 c:\mount
move c:\mount\* c:\templates\
imagex /unmount c:\mount
rmdir c:\mount
```

8 References

Acronis. 2008. Backup software for data backup and disaster recovery in Windows and Linux - Acronis. http://www.acronis.com/index.asp.

Begnum, K., K. Koymans, A. Krap, and J. Sechrest. 2004. Using Virtual Machines in System Administration Education. *Proceedings of 4 thInternational System Administration and Network Engineering Conference (SANE04)*.

Clonezilla. 2008. Clonezilla. http://www.clonezilla.org/.

Corporation, Microsoft. 2008. Updated System Preparation tool for Windows XP Service Pack 2, Windows Server 2003, and Windows XP Tablet PC Edition 2005. http://support.microsoft.com/kb/838080.

Gaspar, A., S. Langevin, and W.D. Armitage. 2007. Virtualization Technologies in the Undergraduate IT Curriculum. *IT Professional* 9, no. 4: 10-17. doi:10.1109/MITP.2007.80.

Lei, Kimfong, and Phillip T. Rawles. 2003. Strategic decisions on technology selections for facilitating a network/systems laboratory using real options \& total cost of ownership theories. In *Proceedings of the 4th conference on Information technology curriculum*, 76-92. Lafayette, Indiana, USA: ACM. doi:10.1145/947121.947139. http://portal.acm.org.ezproxy.rit.edu/citation.cfm?id=947121.947139&coll=portal&dl=ACM&CFID=75240542&CFTOKEN=91715286.

Lowe, Scott. 2008. VD2422: Offline VDI - blog.scottlowe.org - The weblog of an IT pro specializing in virtualization, storage, and servers. September 17. http://blog.scottlowe.org/2008/09/17/vd2422-offline-vdi/.

Microsoft Corporation. 2005. Netsh commands for DHCP: Dynamic Host Configuration Protocol (DHCP); Scripting. January 21. http://technet.microsoft.com/en-us/library/cc787375.aspx.

---. 2007a. Virtual PC 2007 Release Notes. http://download.microsoft.com/download/4/4/c/44ccd131-67fb-4224-a96e-193be1765b43/relnotes.htm.

---. 2007b. Download details: Windows PowerShell 2.0 CTP. November 5. http://www.microsoft.com/downloads/details.aspx?FamilyID=60deac2b-975b-41e6-9fa0-c2fd6aa6bc89&displaylang=en.

---. 2008a. Robocopy. http://technet.microsoft.com/en-us/library/cc733145.aspx.

---. 2008b. Download details: Automated Installation Kit (AIK) for Windows Vista SP1 and Windows Server 2008. April 9. http://www.microsoft.com/downloads/details.aspx?FamilyId=94BB6E34-D890-4932-81A5-5B50C657DE08&displaylang=en.

---. 2008c. Windows Deployment Services Step-by-Step Guide. May 8. http://technet.microsoft.com/en-us/library/cc771670.aspx.

---. 2008d. Windows Deployment Services Processes. May 8. http://technet.microsoft.com/en-us/library/cc753356.aspx.

---. 2008e. Performing Unattended Installations. May 8. http://technet.microsoft.com/en-us/library/cc771830.aspx.

---. 2008f. Using Transport Server. May 8. http://technet.microsoft.com/en-us/library/cc725964.aspx#Common.

---. 2008g. Active Directory Service Interfaces (Windows). August 12. http://msdn.microsoft.com/en-us/library/aa772170.aspx.

Nakajima, Jun. 2007. Hybrid Virtualization - The Next Generation of XenLinux. In . http://www.valinux.co.jp/documents/tech/presentlib/2007/2007xenconf/Intel.pdf.

Sadler, Jez. 2007. transport server : Setup Deployment : Windows Server : Microsoft TechNet Forums. November 7. http://social.technet.microsoft.com/Forums/en-US/winserversetup/thread/ac931db7-efc4-4848-8ade-3aadac1453cd.

Silvert, William. 2001. Modelling as a Discipline. *International Journal of General Systems* 30: 261-282. http://www.informaworld.com/smpp/content?file.txt.

Stackpole, Bill, Jason Koppe, Tom Haskell, Laura Guay, and Yin Pan. 2008. Decentralized Virtualization in Systems Administration Education. In *SIGITE '08*. August 23.

Stockman, Mark, John Nyland, and William Weed. 2005. Centrally-stored and delivered virtual machines in the networking/system administration lab. *SIGITE Newsl.* 2, no. 2: 4-6. doi:10.1145/1072968.1072969. http://portal.acm.org/citation.cfm?id=1072968.1072969.

Sugerman, Jeremy, Ganesh Venkitachalam, Beng-Hong Lim, and VMware, Inc. 2001. Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. In . Boston, MA, June 25.

Symantec. 2008. Ghost Solution Suite: Rapid and reliable Windows Vista migration solution | Symantec. http://www.symantec.com/business/ghost-solution-suite.

VMware, Inc. 2008a. Guest Operating System Installation Guide. http://www.vmware.com/pdf/GuestOS_guide.pdf.

---. 2008b. VMware Workstation Documentation. http://www.vmware.com/support/pubs/ws_pubs.html.

---. 2008c. VMware ACE Enterprise Desktop Management, Virtual Machines - VMware. http://www.vmware.com/products/ace/.

Vollrath, Adam, and Steven Jenkins. 2004. Using virtual machines for teaching system administration. *J. Comput. Small Coll.* 20, no. 2: 287-292. http://portal.acm.org/citation.cfm?id=1040189&dl=GUIDE&coll=GUIDE&CFID=34707486&CFTOKEN=32122091.