

U.S. NAVAL ACADEMY
COMPUTER SCIENCE DEPARTMENT
TECHNICAL REPORT



Virtualization Shares: Feasibility and Implementation in the USNA
Computer Science Department

Christopher Wheeler

USNA-CS-TR-2010-03

March 10, 2010

Computer Science Department
IT495B: Research Project Report
Fall AY09

**Virtualization Shares: Feasibility and Implementation
in the USNA Computer Science Department**

by

Midshipman Christopher Wheeler, 107038

United States Naval Academy
Annapolis, MD

Date

Certification of Faculty Mentor's Approval

Assistant Professor Thomas Augustine
Department of Computer Science

Date

Department Chair Endorsement

Professor Donald Needham
Chair, Department of Computer Science

Date

Executive Summary

The research study, *Virtualization Shares: Feasibility and Implementation in the USNA Computer Science department* was conducted at the United States Naval Academy in an effort to help define a how sharing virtual machines which had been transferred via external hard drive from host to host, and run on VMware workstation, could be run on a single powerful server and require users to interact with them using a thin client. Specific topics cover basic virtualization concepts, differences in architecture between Xen and VMware, and the performance seen on a test network utilizing one server running ESX.

As corporations and other large enterprises, including the Department of Defense, move from the traditional physical server infrastructure towards virtual consolidation, study in this area becomes more and more pertinent. In the USNA Computer Science Department, this server resides on a sandboxed network, used only for testing purposes, but this technique has been implemented across many major organizations running servers as a result of low utilization of traditional physical infrastructure. Using a virtualized architecture allows more dynamic load sharing based on the current demands placed on a particular host, and overall results in less idle time on the infrastructure.

The goal of this research paper was to define potential architectures that satisfy our existing needs, including labs for Information Assurance classes, exercises such as Cyber Defense Exercise, and development work. By analyzing their relative performance, a compromise between performance, ease-of-use, and the resources of the Department provided recommendations that will become an integral part of Computer Science and Information Technology education.

At the conclusion, numerous studies on both VMware and Xen architecture were analyzed, which gave insight into architectures to be modeled by the Department. For the purposes of research, Xen was focused on more heavily by nature of being open source. However, our current VMware license weds us to their infrastructure, the main reason for solely analyzing ESX. This study may also lead to further research into topic areas such as dynamic image swapping across multiple servers, vulnerabilities of virtualization shares, and even more utile architectures for the Department's needs.

1. Introduction

As the cost of producing advanced server hardware with greater-than-necessary capability continues to decrease, virtualization has become even more crucial and widely utilized in server operations. Rather than let this power go unused, virtual server environments seek to provide many environments on a single or small number of servers, with each virtual server serving a distinct purpose. In the case of the Naval Academy's computer science department, these hosts function as experimental machines for Information Assurance classes, which act on both the offensive and defensive sides of dll injections, port scanning, and other basic hacking techniques. My goal in this paper is to examine the feasibility of establishing a shared server for these images, so that rather than copying a prepared image to each individual host either through FTP or various hard drives, they could be modified on a single host, uploaded to the server, and have groups of Midshipmen modify them simultaneously. In the worst case, the share would be configured as usual, but copies of images would be "pushed" to or "pulled" from hosts running a compatible client.

2. Literature Review

There are two types of virtualization: process and system. "Process virtualization" is barely regarded as such in the sense the term is used today, as it has become a requirement in kernel operations, and goes unnoticed in everyday computing applications. In process virtualization, each process is allocated its own address space, registers, and file structure. Albeit small, this constitutes a virtual environment, especially when a critical system process is permanently assigned these attributes, much like a chroot jail assigns a process a smaller filesystem. "System virtualization," the primary focus of this paper, is when a relatively powerful *host* allows *guest* operating systems to run within the existing operating system (Smith and Nair, 2005). On any given machine, a certain instruction set architecture (ISA) is utilized by a user through the operating system to achieve desired effects from the machine's hardware. Because Microsoft Windows' ISA does not mesh nicely with that of a Debian Linux variant's, virtualization software such as VMware serves as an intermediate to allow a virtual image of one operating system to run inside the other. This software allocates memory, processor time, and bandwidth among other limited performance factors from the host itself to the virtual machine(s) running within. Examples of this software for a single host include VMware[®] Workstation, and Sun[®] VirtualBox.

One of the prime experimental mediums for this study will be VMware ESXi server. ESXi and ESX are known as Type-1 *hypervisors*, which means they run directly on a powerful host's hardware and act as a monitor and a share for the virtual hosts that run on them. Both sets of software contain what will be referred to throughout the course of this study as virtual machine monitors (VMM's), to control various parameters crucial to virtualization. On any virtualization server, the hard disks of the host are partitioned into separate shares for each hosted VM (Virtual Machine).¹ For the purposes of this study, this powerful host will be a Dell PowerEdge 2950 Server² running VMware ESXi.

1 "VMware ESX and VMware ESXi." Brochure. Available at: <http://vmware.com/>

2 Specifications listed in Appendix B

Another Type-1 hypervisor is Citrix® XenServer, which has a much lower fingerprint on the server it runs on than VMware's ESX. In an ESX environment, the guest host is "unaware" that it is being run virtually, which requires proprietary VMware drivers for each server component that the VM's require access to (i.e. Network interfaces, storage configurations). XenServer, on the other hand, makes it obvious to the guest OS that it is being run virtually and interfaces with the Xen® management console, known as "Domain 0," which then interacts with the hypervisor through open source drivers. "Domain 0" is actually a separate VM running a hardened and optimized Linux kernel. This process is known as *paravirtualization*, which works almost seamlessly for Linux OS's on both the VMware and Xen hypervisors. Windows operating systems, by contrast, cannot be fully paravirtualized, and for certain instructions rely on the host hardware's virtualization assist technologies.³

The primary areas this study will focus on that we expect to limit our setup's capabilities are disk input/output (I/O) rates, virtual memory allocation, virtual network interface card (NIC) availability, and central processing unit (CPU) sharing, in increasing order of effect. Many methods of improving virtualization sharing have been put forth in various studies.

One such method is VMM I/O bypass, in which interactions with devices themselves go through a separate, specially designed VM, vice the VMM. By eliminating the VMM in this operation, an area of bottlenecking is eliminated, and may help the spread the load across different platforms. Unfortunately, while this is possible using the Xen architecture, which is based heavily on paravirtualization, ESX requires that all I/O operations go through the VMM (Liu 2006). As seen in Fröning's study, this serves as a limiting factor in performance, as VNI's require extensive I/O operations (Fröning 2009).

A proposed way to drastically reduce this overhead is through concurrent, direct network access (CDNA). As compared to the traditional infrastructure, in which the each VM interacts with a separate back-end driver in the hypervisor, CDNA utilizes a single CDNA NIC, which all the VM's interact with and the hypervisor still monitors. Versus traditional Xen and VMware implementations, this yields performance improvements of 370% transmit throughput and 126% receive throughput in a 24 VM setup. As of 2008, this has yet to be implemented in either of the two vendors' products (Rixner 2008).

This network and I/O overhead is a key concern when considering whether purchasing an ESX infrastructure will be worth its cost to the Department. A June 2009 study on parallel computing using VM's on an ESX infrastructure found that the I/O limitations of the setup yielded significantly slower runtimes when more than 32 VM's were utilized simultaneously (Martinez 2009). This was mainly due to inter-communication overhead both on the VNI's and in the hypervisor. This setup, however, used processes that were continually communicating and executing. For the purposes of the department, our traffic and infrastructure utilization will come in spurts as computers are scanned, attacks are executed, and defenses are put in place, so our performance should be significantly better.

3. Test and Analysis

3 "Technical and commercial comparison of Citrix XenServer and VMware." Available at: <http://citrix.com>

This study seeks to find an efficient way of creating a virtual setup, usually of one to three virtual machines on a bridged adapter, modified to fit the purposes of an Information Assurance training lab, and propagating that setup to a cluster of hosts. None of the hosts will be paravirtualized to ensure reliable test results across virtual platforms. If necessary, the setup could be “pull” based, meaning the host(s) would connect to the server and request the set of images through VMWare software. If this is infeasible due to licensing costs for additional software, or due to bandwidth restrictions, the server could simply be utilized as an FTP share. Where ESX truly has great potential, however, is the simultaneous modification of images running on the server. However, this would require more coordination between lab partners, and rewriting most of the labs for Information Assurance and Advanced Information Assurance, as they are based on single host setups.

In order to test these requirements, this study tests four major aspects. First it tests the use of linked clones running off a Samba file share, followed by using the server as a file share for images, then moves into more advanced architecture provided by vSphere. This entails simultaneously modifying the images, and testing their performance using key metrics such as memory consumption, virtual CPU strain, and network bandwidth.

To minimize our test environment, configurations will first be tested with a single host, then with four hosts (ref. Figure 1). If a test run shows a great deal of promise, it can be run again with the full sixteen hosts. The overall assumption for our original configuration, with an instructor preparing images on another workstation and transferring them using a USB hard drive, is that movement to each host takes roughly 13 minutes.

$$(13 \text{ min}) / (2 \text{ hard-drives}) = 7.5 \text{ minutes/ image} + \text{overhead of moving around}$$

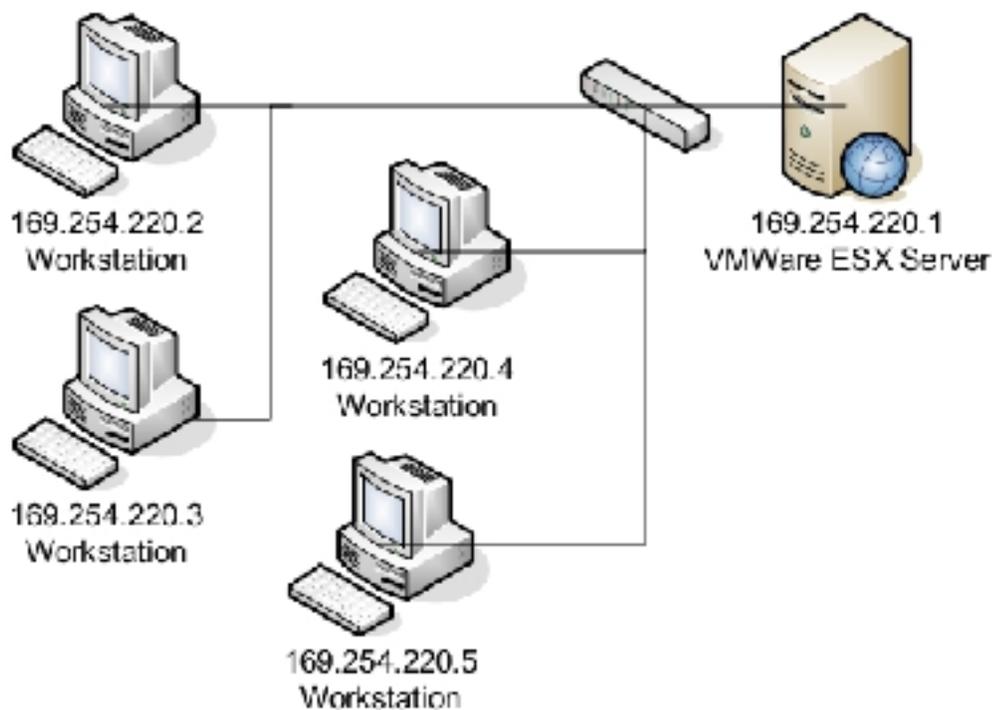


Figure 1: Test Setup

Test Procedure 1: Using the existing CS Department license for VMware workstation, linked clones were created and run off the “Tera” server.

Details: Tera is a 1 terabyte Samba (SMB) fileshare, and one exists for both Michelson lab 221 and Michelson lab 201. A linked clone of a virtual machine means creating a manifest of the changes to the virtual disk, while referencing the original disk at the time of the clone as the primary hard drive. Thus, the linked clones are very small files if they have not been modified, and would be easy to push out to clients. For this procedure, we used the Tera server in Michelson 201, had 8 students map the Tera server, open prepared linked clones of a Windows XP image in VMware workstation, and booted them simultaneously.

Findings: This test failed entirely. Students experienced over 10 times the normal boot time for the image, and some failed to read the original virtual disk entirely once others had booted the image. From a Wireshark capture, it appeared that the server, accessed so many users at the same time, sent traffic out to its clients in spurts. This can be attributed to slow disk speeds on the Tera drive and the way the Samba protocol responded to multiple simultaneous clients reading from a shared drive, which generated unbalanced load sharing. The test was scrapped before all but one image could open a user’s

'Desktop,' as the time to configure the virtual network of the images would have taken additional time.

Conclusion: Without an incredibly expensive infrastructure which allows for extremely fast disk reads and writes, or a more efficient file sharing protocol, this setup is not feasible.

Test Procedure 2: Virtual machine created on ESX host through a vSphere client, transferred over FTP by cluster of hosts using the same client.

Details: Clients simply browse the datastore (this would require another user, for example 'iastudent,' but for purposes of testing, 'chris,' a member of the local administrators group was used), and this user saved the entire 'Backtrack 4' directory to 'Desktop.' The vdmk hard drive (the main source of any delay, as configuration and log file sizes are essentially negligible) is a flat 5 gigabyte virtual disk.

Findings: Data transfer rate ceilings at roughly 45,000 Kbps (ref. Figure 2), causing the transfer to take roughly 15 minutes. On the single host test, results were comparable, as the test took roughly 4 minutes.

$$(15 \text{ min}) / (4 \text{ hosts}) \times (16 \text{ hosts}) = 60 \text{ minutes/image}$$

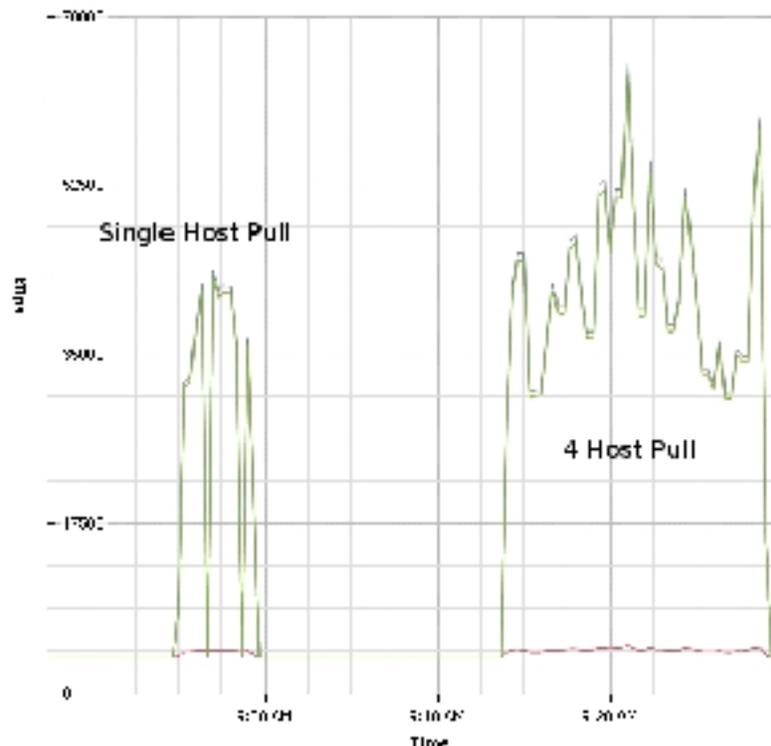


Figure 2: Data Transfer Rates on ESX host

Conclusion: While this doubles the speed of the existing procedure, it still does not meet our requirements. The process is still quite slow, even when using a switch instead of a hub. Additionally, if administrator access is required to access these images, the

instructor would have to log them off upon completion, so they couldn't just hit download and leave the lab. This assessment also doesn't consider the case where the size of the hard drives is greater than 5 GB; using simple math it would increase the time to get the images out by $(([size\ in\ GB] / [5\ GB] - 1) * 100)$ percent, without factoring in overhead.

Test Procedure 3: Virtual Machine created on ESX Host through vSphere client, pushed or pulled by clients at one time. This will be considered to be negligible in the future, because the transfer analyzed in this test will be that of *linked clones*. Linked clones are essentially a track record of changes made to a virtual machine, so over time, the one time cost of transferring the image to the host becomes negligible.⁴

Details: Same setup as above. The current hang-up with this setup is that the department lacks a full license for vSphere, which limits a user from creating a clone. The ESX UI on the host itself does not include a feature to push the images.

Findings: Since VMware workstation and ESX are not “out-of-the-box” compatible, the current method of placing an existing image on the ESX server has been to create a fresh image of the same OS on the server, and instead of creating a new virtual disk, adding the existing virtual disk as the hard drive. This often requires a simple hack to fool the newly created machine into believing the hard drive is native to it: naming everything exactly as it was in the original image. This has been of particular concern for Windows images whose hard drives have been split into 2GB chunks. This manner of partitioning requires numerous extra configuration files, which must be copied into the correct directory under the proper name in order to function properly. Linux hosts, especially those with “flat” virtual storage allocations, are not problematic at all. Currently, this setup could be used as a backup to other methods, but would not be fully utilizing the expensive vSphere license.

Test Procedure 4: Virtual Machines will be modified simultaneously, and the labs for each class or setup the virtualization share would be used in would be rewritten to fit this model.

Details: There are roughly 10-12 labs for each IT430 and IT432 class. Teams would consist of 4 people, or however many sat at one table. Stress testing for the server consisted of running 6 hosts on the server (Backtrack, 2-Windows XP, Ubuntu 9.10 beta, and 2-Ubuntu 9.04 servers), ping floods from all Linux guests, intense port scans from Windows guests, and infinite ‘while’ loops on all hosts.

Findings: In the ideal case, while users end up “fighting” for the keyboard, it encourages collaboration. If one group member is more knowledgeable than the other, he can do what he needs to to the image, and say out loud to the group what he is doing, thereby instructing everyone at the table without students having to look over one another’s shoulder. Running them on the server itself eliminates the “push” or “pull” from the

4 Procedure detailed in Appendix A

clients all together, so time to execute the lab is negligible. Stress testing the server yielded no noticeable changes to the end user, but CPU usage on the ESX server spiked. The only time considerations in this setup are the preparation time, and changing the labs to meet this new method of practicing IA. The former has never been considered in this analysis, and therefore will not count toward the time. Re-writing the labs, however, is a serious time consideration with great potential.

From our testing of an Advanced Information Assurance lab involving three images, users were not as keen on the collaboration as originally expected. Likely, this was a result of precedent, since for one and a half semesters of Information Assurance classes, individual images had been provided to users on each host. The day of the test, and only minutes before they started the lab, this new approach of simultaneous modification was introduced to the students, which may have been the reason for their confusion.

Where this setup holds the greatest potential is a hacking competition against a particular target image. Access permissions on the target could be limited to the administrator or instructor, so users could not open a console and modify the image. Better yet, they could be set as view-only, so that the user could see what effects their attacks had on the box, without any ability to execute programs on the image. If a simple script which ran 'netstat' in a time loop was used on the target image along with a process listing, the end user could also view how stealthy any connections they made to the box would be, and this host could even be projected up onto the screen so all teams could monitor their progress. In this case, however, virtual hardware availability becomes a concern. Consider RAM:

$$\begin{aligned} & (4 \text{ teams}) * [(2 \text{ Backtrack images}) * 256 \text{ MB} + (1 \text{ XP Image}) * 512 \text{ MB} \\ & \quad + (1 \text{ Ubuntu Image}) * 384 \text{ MB} +] \\ & \quad + 1 \text{ Target Image} * 512 \text{ MB} \\ & \quad = \sim \mathbf{6144 \text{ MB of RAM}} \end{aligned}$$

Values based upon VMware recommendations and IT430/2 precedent

8192 MB is the available RAM for the current setup, and since ESX idles around 800 MB of usage, only 7392 MB is available.

Virtual CPU's are another consideration:

$$4 \text{ teams} * 4 \text{ images} + 1 \text{ target image} = 17 \text{ images} \rightarrow \mathbf{17 \text{ Virtual CPU's}}$$

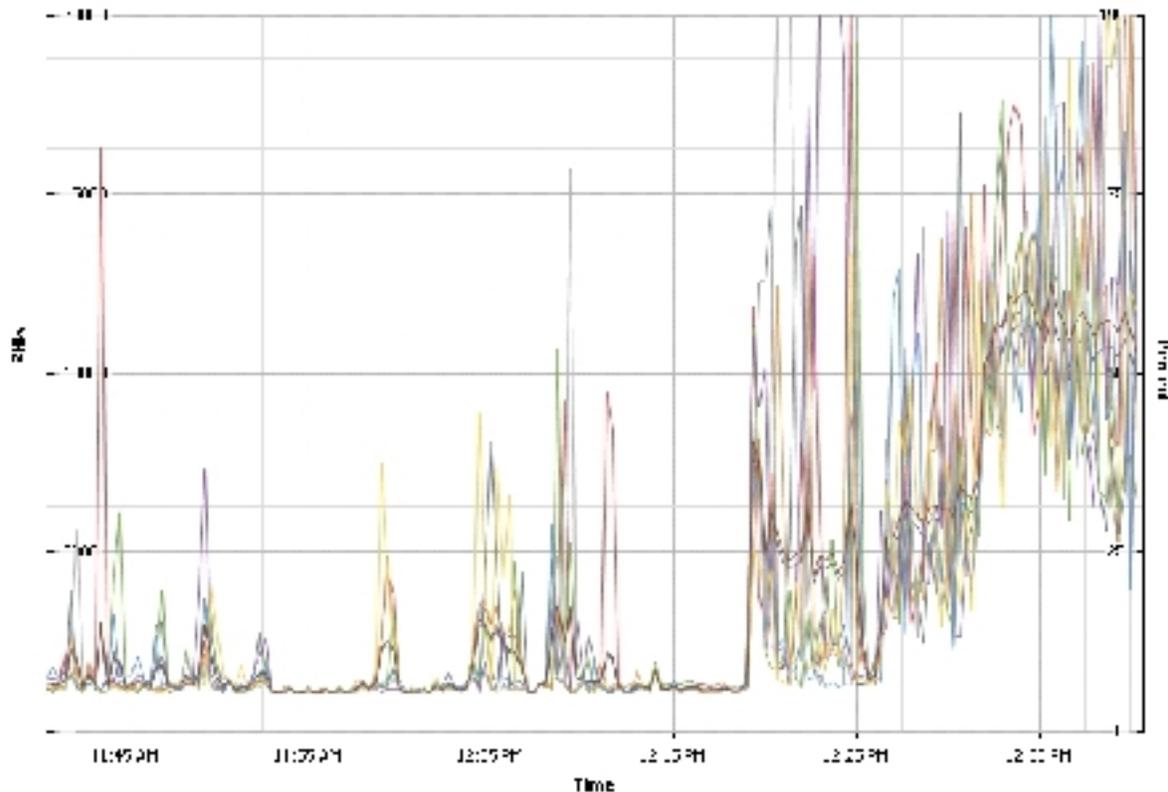


Figure 3: CPU Performance During Stress Testing

With the current setup, there are only 8 CPU's available. Further dividing them to fit this lab setup will decrease performance and greatly increase stress on the server (ref. Figure 3). Even though ESX allows this, the swapping to allocate CPU time to a given host does not allow a full processor to be allocated to each VM in the case where all 4 teams are using half of their images, or vice versa. This will be examined more methodically later in the analysis.

Test Procedure 5: Open Virtualization Format (OVF) implementation. The template would be deployed to the whole server, and should allow for compatibility between the Linux-based ESX server and the Windows sandbox machines running VMware workstation.

Details: OVF provides more data about a machine than a vdmk virtual disk can, to include memory allocations, CPU allocations, display name, referenced configurations files, device nodes, and network configurations. It is a complete specification of the virtual machine or set of virtual machines. From vSphere, this is as simple as a point and click. Potential exists in packaging our current images or lab setup into this format, so that it could simply be uploaded to the server and run there, or run from an active administrative client with the OVF file on the local disk.⁵

⁵ http://www.youtube.com/watch?v=Ob3UVPII7mE&feature=player_embedded

Findings: While supposedly cross-platform and cross-virtualization-software compatible, this is not the case. The best example was attempting to port fresh VirtualBox images to VMWare, or vice versa. Even without their respective toolkits installed, these OVF templates have a host of problems: not being able to set configuration parameters, not being able to recognize the hard drive, even with the proper file vdmk file extension, and not being able to recognize the OVF configuration file because of different namespaces and the number of device nodes supported. What was helpful about VMware to VMware OVF transfers was that it generally compressed the hard drive, reducing file transfer time. If an FTP-based solution was required, this would serve as an excellent way of packaging the images.

Test Procedure 6: Limited resources are an ever-present problem with virtualization software. Considering that our ultimate goal for this server is to run a large number of images to support a class or competition, two key areas of consideration come to mind: RAM and CPU limitations. This test procedure focuses on CPU overloading, which may become problematic when each machine is allocated a set number of virtual CPU's, and the sum of allocated virtual CPU's becomes greater than the number of physical CPU's available.

In order to test the effects this would have on our current setup, we booted all the virtual machines from our 'Cyber Weekend' resource pool, totaling 12 virtual machines, each with a single virtual CPU allocated. Our current ESX setup contains 4 CPU's with 2 cores each, for a total of 8 possible dedicated CPU's, assuming simple load sharing practices. With ESX acting properly, we would anticipate that load sharing and CPU swapping would occur. Upon boot, we had some spikes, up to 75% on a single CPU, but after settling into boot mode, they idled under 5%.

To introduce extreme loads on the CPU, we put every virtual machine in an infinite loop:

Windows:

```
|| infinite_loop.bat  
  
@@ ECHO OFF  
:l  
echo "test"  
goto l
```

Linux:

```
#!/bin/bash  
  
while true; do  
echo "test";  
done
```

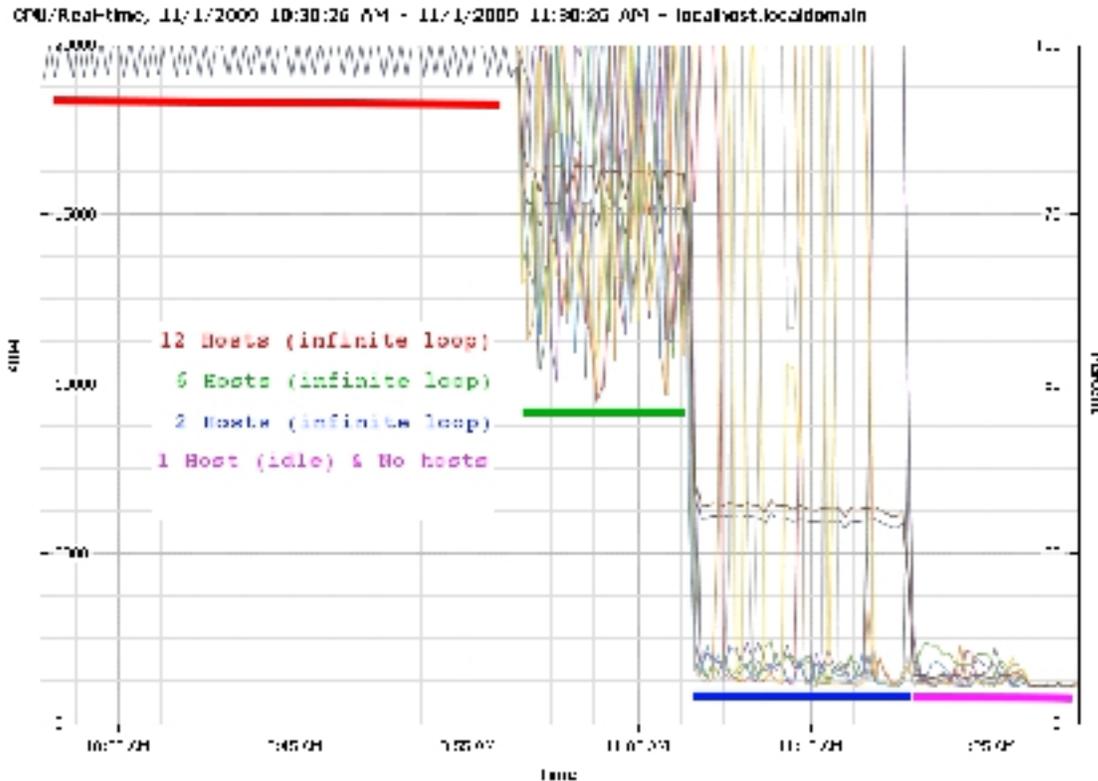


Figure 4: CPU Performance during Infinite Loop Stress Testing

Although it functioned properly in our stress testing, this is obviously not a sustainable mode of operation without a much more reliable cooling mechanism for the server (ref. Figure 4). Our earlier test setup, where CPU usage gradually ramps up, is a much more realistic scenario.

The fact that two VM's were booted while all system CPU's were running at close to capacity leads me to conclude that that the load sharing in ESX server may allow the server to temporarily overcome limited resources, but it is not desirable when any sort of performance is required.

Test Procedure 7: In order to see the effects of over-committing our server's memory resources, we ran a similar setup to the CPU test, we ran 4 guests at 2GB of dedicated memory each, which should take our memory over the edge and generate memory sharing or extensive paging. Each VM was allotted a single virtual CPU, as in our last setup, so as not to make virtual processors a limiting factor.

$$4 \text{ Virtual Machines} \times [(2048\text{MB allocated memory}) + \sim 100 \text{ MB Overhead}] = \sim 8592 \text{ MB Allocated}$$

$$8592 \text{ MB} > 8192 \text{ MB physical server memory}$$

When we added a test machine, a Windows XP with 256MB of RAM, with 4 others allocated above the memory threshold, there was no effect on the machines

execution, which appeared to be a result of ESX independently adjusting our allocations. Allocating 8192 MB of RAM, the “granted” memory for the system idles at 7,500,000 Kb, or 7324 MB. To ensure we went over the threshold, we then started booting machines.

This has no effect on ESX. In order to ensure the accuracy of our findings, I continued to boot virtual machines, and ran memory intensive processes on all of them, to no avail. The only time an impact was observed was when I ran simple programs on guests to infinitely allocate memory in chunks:

add_memory.c

```
#include <stdlib.h>
int main(){
    while (1)
        malloc(1024);
    return 0;
}
```

This affected both the server’s performance monitor and that of the VM itself. Otherwise, from the server’s performance monitor, it simply showed that the server had allocated all its available memory (~7500MB, with overhead = 8192 MB).

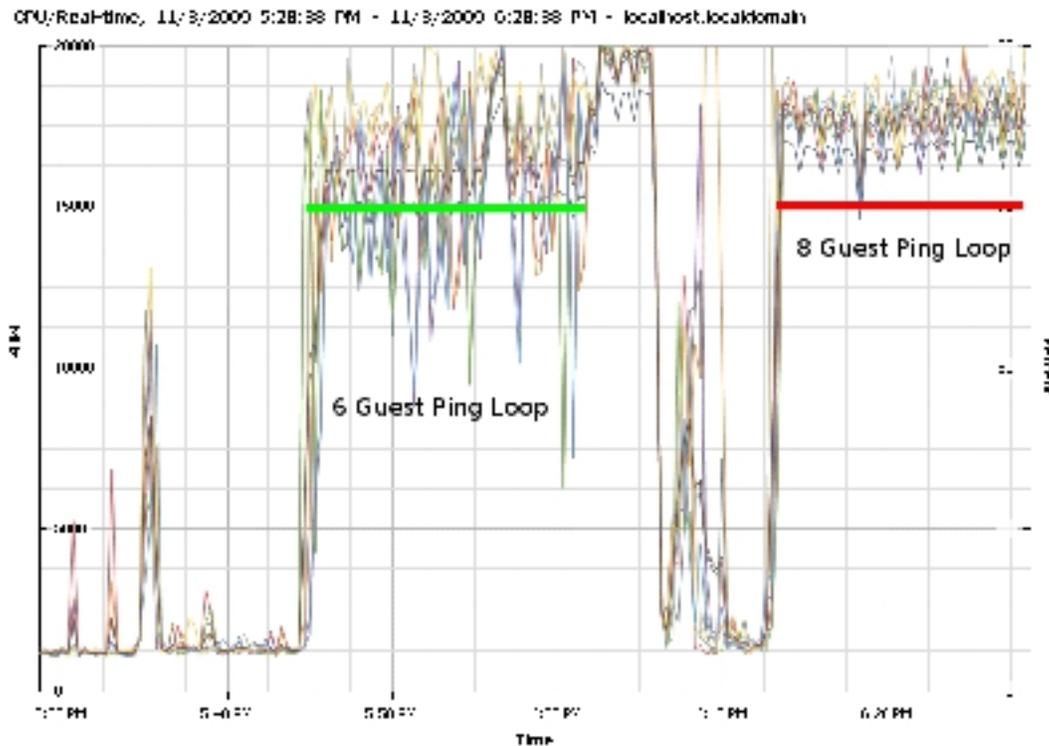
This result is validated by Waldspurger’s study as VMware has implemented since early versions of ESX the ability for memory to “balloon” based on the immediate requirements by the VM’s at any given time. Many algorithms have been developed to efficiently manage this sharing by VMware and independent scholars alike, which is one of the reasons for ESX’s wide deployment. Another is that this feature distinguishes it from VMware Workstation, which has a policy of not being able to over-allocate memory without informing the user; the dialog in Workstation reads “memory swapping may occur, reducing performance.” With ESX, memory, CPU, and resource swapping in general is considered implicit, hence its built-in design to manage such loads (Waldspurger 2009).

Test Procedure 8: With CPU and RAM being efficiently managed by ESX, another remaining factor was how network traffic affected the host.

Details: 6 VM’s, all Linux based, were booted and each allocated a single Virtual CPU and 512 MB of RAM. Later, the number was expanded to 8, but to ensure Virtual CPU’s were not a limiting factor, 6 were used for our initial test. IP’s were statically set in increments of 10, and each host ran a ping flood on the subsequent host in the chain (i.e. 10.10.10.20 pinged 10.10.10.30, while being pinged by 10.10.10.10). This generated network bandwidths of 200-300 KB/s, where the host did not limit the number of outgoing packets. This test yielded the most curious results of all: on the process monitor within the VM, little to no CPU activity was observed. On the server’s performance monitor, however, the CPU’s appeared to be maxed out, and even over-maxed on those hosts that were running an unlimited ping flood.

Findings: VM performance slowed on some machines, especially once they began both sending and receiving packets. When 8 hosts were run in the same ping loop, the problem became worse. While no official “benchmark” was taken, the test I ran started

with simple operations, such as opening a terminal, then moved on to more complex task, like opening a web browser, then the openoffice.org suite. The delays experienced appeared to be related to X rendering; when a terminal was opened, first a white box would appear, the colors would render, then a full shell would appear. This, on average, took 3-5 seconds. Normal operation is almost instant. Typing delays occurred, as well, which hovered between 0.5 and 1.0 seconds just to type into the web browser and the terminal. These tests were run on Linux with a system monitor up, and the CPU experienced no spikes. ESX's view of the operation was much different:



This likely spawns from the setup's lack of physical NIC's for each VM, which is a common problem to any virtual architecture. As evidenced by the guests' system monitor view of the CPU time, the ping flood itself is not taxing to the CPU. However, the physical CPU's, which are assigned as Virtual CPU's, have to process all incoming and outgoing network packets. In the scenario we demonstrated, this was especially strenuous because all the guests were both sending and receiving, and in the case of Ubuntu, rendering a robust X window, which on a physical machine would be handled by a separate GPU.

Fröning's study backs this hypothesis. Virtual network interfaces (VNI), by design, require more work with respect to the virtual CPU assigned to the machine, since the same CPU needs to analyze which processes are sending data across the network. More specifically, it needs to package the data, and schedule when these packets can go across the network. VNI's, like any network interface card, require a queue, but the fact that a virtual PID has to be inserted into this queue in addition to the data itself means that it requires a read to determine whether space exists on the queue, and a write to

actually insert the data. Combined with overhead, all these factors make the VNI significantly less efficient than a physical NIC, and account for the extreme stress on the CPU's when 6 to 8 VNI's are sending and receiving simultaneously (Froning 2009).

4. Summary and Conclusion

Based on the above tests, the most problematic area of the network is virtual CPU availability, since it plays a role in so many operations that are otherwise handled by physical hardware. This is especially evident with the VNI, a crucial part of the setup considering that extensive traffic will be passed across the network to scan, exploit, and deny service to hosts by many users simultaneously. While the load will rarely be as focused as the one experienced during stress testing, more users than the number used for testing will be using the images on the server, many of which will be doing RAM, CPU, and network intensive processes all at one time as a result of the nature of the lab or exercise. Since two out of three of these potential bottlenecks are handled by the CPU's, the setup purchased by the department should contain as many cores as possible, while minimizing the number of actual processors to cut down on licensing costs (ESX is licensed by processor, independent of the number of cores). As more cores become available, more physical CPU's that can be assigned to a virtual machine in a case when there are less images than [cores] * [processors], and the more load sharing that can be enabled if not.

5. Recommendations

Based on this study, I recommend the Department continue expanding the ESX infrastructure. Because of the overwhelming stress on CPU's during intense network operations, which will be crucial to Information Assurance and Computer Networking courses, the Department should purchase servers with more cores per socket vice overall number of cores. Since ESX licensing costs are based on sockets, not number of cores, a contest between a server with two sockets and six cores per processor, and four sockets and four cores per processor would undoubtedly go to the two socket server. ESX licenses cost roughly ~\$1100 per year per socket as of the time of this study, so our current setup would cost us roughly \$2200 per year to fully license.

Another recommendation I put forth is that the Department purchase more, less-expensive ESX servers vice fewer servers with top-of-the-line specifications. Aside from limiting single points of failure, this also provides opportunities to experiment with and study technologies such as VMmotion, in which running images can be migrated from server to server in real-time. This study can be geared toward performance or the security of such technologies, both of which are growing concerns in modern Department of Defense and production environments.

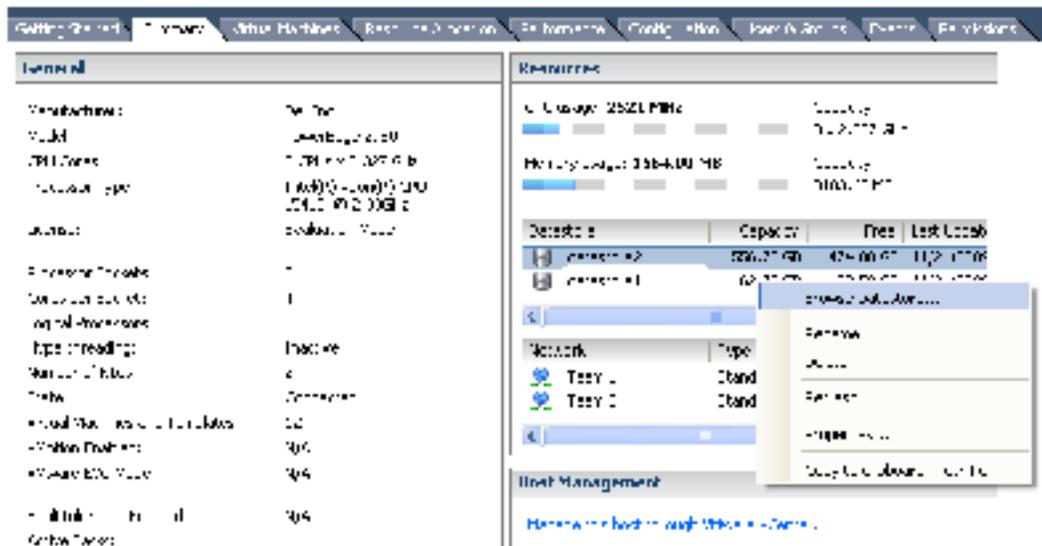
My final recommendation is broader in scope: when the servers are purchased, each Information Assurance lab should be outfitted with dedicated servers, and the two sandbox networks should be connected to allow for larger scale exercises utilizing the infrastructure from both labs. ESX provides an excellent framework for dynamic, real-time exercises to be conducted, which is not a capability of the pre-formulated labs we

conduct in our current Information Assurance classes. These dynamic exercises will more accurately portray the situations our Information Professionals, Information Warfare Officers, and graduates in general are likely to encounter in the Fleet and beyond.

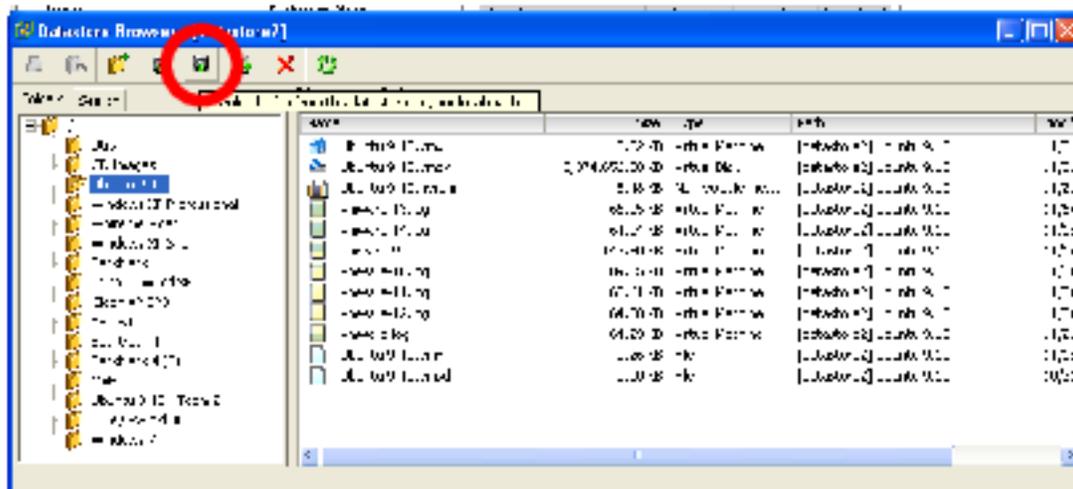
Appendix A: Uploading and downloading images from an ESXi server

1. Download using vSphere client

- a. Connect to the ESX server by clicking 'vSphere Client,' entering the IP, username, and password
- b. Select the server (192.168.1.200)
- c. Under the 'summary' tab, look on the right side, and click whichever datastore the image is on:



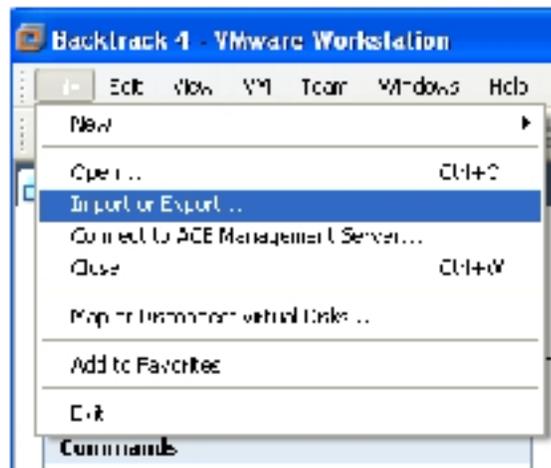
- d. In the left hand column of the 'datastore browser' dialog, click on the **folder** of the image you wish to download, and select the download button



- e. Select the destination folder you wish to put the image into
- f. Click next, and download. The images folder will be available in the directory you specified.

2. Download using VMware Workstation

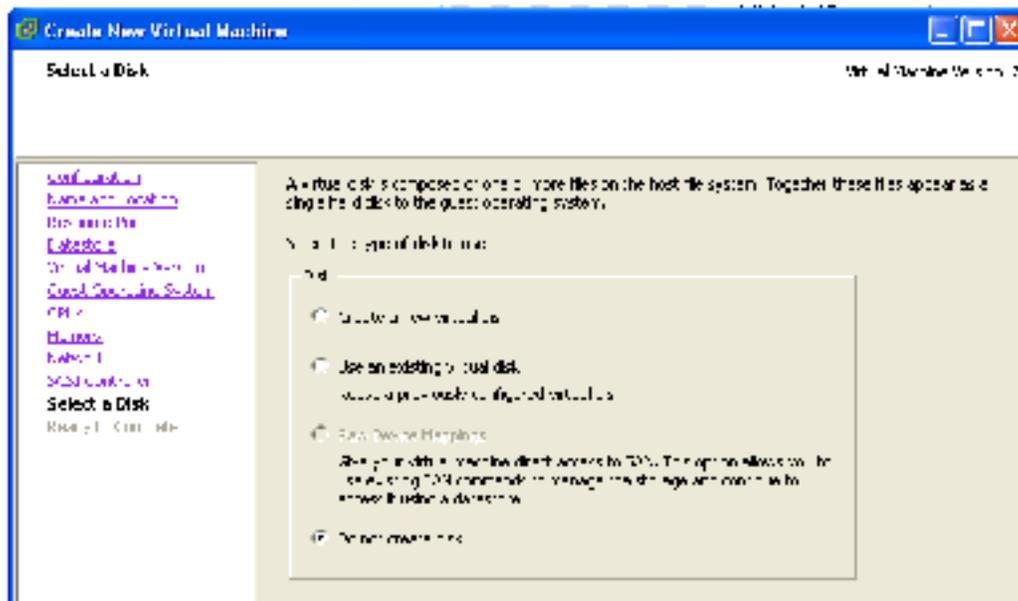
- a. Open VMware Workstation, and from the file menu, select import/export



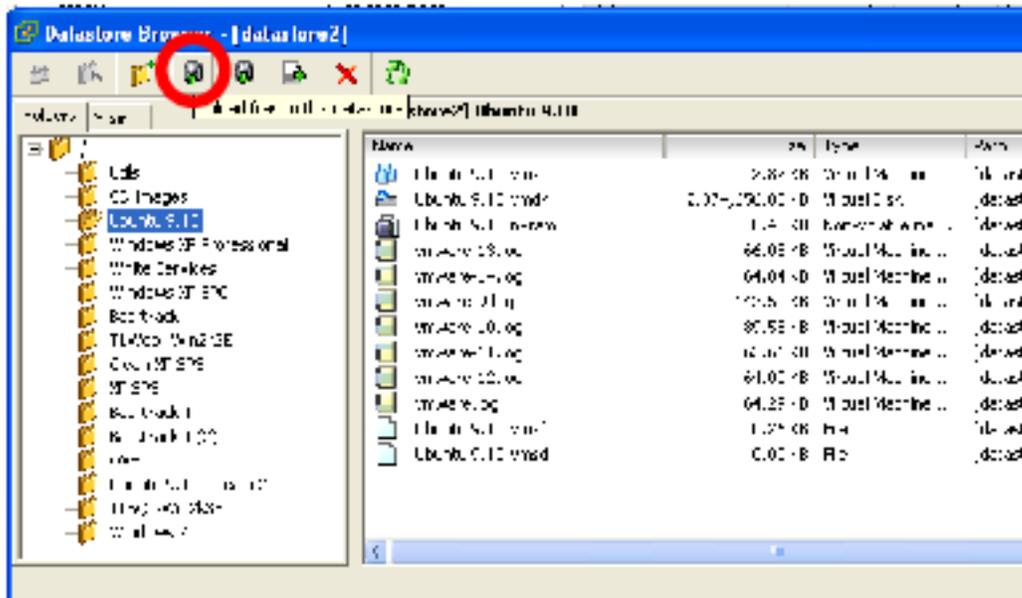
- b. Select 'VMware Infrastructure Virtual Machine' from the combo box, and press 'Next'
- c. Enter the IP address, username, and password as if you were connecting using vSphere client, press 'Next'
- d. Select the image you wish to download and click 'Next'
 - i. *Note: an error message will likely appear stating 'Cannot configure source image' – Ignore this*
- e. Click 'Next' until you reach the destination type combo box- select 'Other Virtual Machine,' and hit 'Next' again
- f. Name the Virtual Machine, and select the destination directory by using the 'Browse' Dialog
- g. 'Allow virtual disk files to expand' will be selected by default- leave this and hit 'Next'
- h. Continue to hit 'Next' through the NIC configurations until the image starts to download. This image will be automatically available to be opened in workstation and will reside in the directory you specified.

3. Upload using vSphere client

- a. Log on to the ESX server as specified in part 1a
- b. Right click on the server, and select 'New Virtual Machine'
- c. Select 'Custom'
- d. Specify parameters of your VM as you would in VMworkstation
- e. When you reach the 'Select a Disk' dialog, select 'Do not create disk'



- Hit 'Next' and complete the virtual machine.
- Browse the datastore you created the machine in, as detailed in part 1c
- Select the '/' directory your VM resides in on the left hand side of the dialog, and select upload



- Select the vmdk files from the directory of the image you are trying to upload, and hit 'Next' until it uploads
- In the main vSphere client window, right click the virtual machine you just created, and select 'Edit Settings'
- Under the 'Hardware' tab, select 'Add'
- Select 'Hard Disk', then 'Use an existing virtual disk'

- m. Browse to the location of vmdk file in the datastore, and click 'Next' until it is added.
- n. Boot the virtual machine by right clicking it, then Power → Power On
- o. View it by right clicking on it again, and selecting 'Open Console'

4. Upload using VMware Workstation

- a. Open VMware Workstation, and select 'Import/Export' from the 'File' menu as detailed in part 2
- b. Select 'Virtual Appliance' from the combo box
- c. Select File System, and browse to where the image is stored on your physical hard drive
 - i. *Note: a message will appear here stating that 'The image specified is a backup image' – click OK*
- d. Click 'Next' until you get to specifying the destination type.
- e. Select 'VMware Infrastructure Virtual Machine,' and log in as detailed in part 2c
- f. Select the datastore and resource pool to add the machine to, click Next
- g. Click 'Next' through any error messages
- h. Once uploaded, boot your machine on the ESX server by right clicking on it, then Power → Power On
- i. View it by right clicking on it again, and selecting 'Open Console'

Appendix B: Dell PowerEdge 2950 server specifications

Model: Dell Poweredge Energy Smart 2950 III

Processor: Quad-core Intel Xeon L5410 – 2x6 MB Cache.
1333 MHz Front-side Bus

Memory: 8GB Memory @ 667MHz (4x2GB)
Dual Ranked DIMM's

Storage: Primary Hard Drive: 2.5", 73GB 10K RPM Serial Attached SCSI
-3GBPS, SATA
Secondary Hard Drive: 450GB 2.5" SATA. 10K RPM.

RAID: Primary Controller: PERC6I SAS RAID Controller, 2x4 connectors, 256MB
Cache

Network: Dual Embedded Broadcom NetXtreme II 5708, Gigabit Ethernet

CD-Drive: DVD-RW

Works Cited:

- Froning, Holger, Heiner Litz, and Ulrich Bruning. "Efficient Virtualization of High-Performance Network Interfaces." *Networks* 1 (2009): 434-439. <http://ieeexplore.ieee.org> (accessed October 8, 2009).
- Lui, Jiuxing, Wei Huang, Bulent Abali, and Dhableswar Panda. "High performance VMM-bypass I/O in virtual machines." *USENIX Annual Technical Conference* 1 (2006): 3. <http://portal.acm.org/> (accessed October 23, 2009).
- Martinez, Juan, Lixi Wang, Ming Zhao, and Masoud Sadjadi. "Experimental study of large-scale computing on virtualized resources." *International Conference on Autonomic Computing* 1 (2009): 35-42. <http://portal.acm.org/> (accessed November 1, 2009).
- Scott, Rixner. "Network Virtualization: Breaking the Performance Barrier." *ACM Queue* 6, no. 1 (2008): 39-45. <http://portal.acm.org/> (accessed November 7, 2009).
- Smith, James, and Ravi Nair. "The architecture of virtual machines." *IEEE Explore* 38, no. 5 (2005): 32-38. <http://ieeexplore.ieee.org> (accessed September 3, 2009).
- Waldsburger, Carl. "Memory Resource Management in VMware ESX Server." *OSNI* 1 (2002): 1. <http://www.usenix.org/publications/library/proceedings/osdi02/index.html> (accessed November 14, 2009).