2015

Virtualization Components of the Modern Hypervisor

Sean McAdams
University of North Florida, n00172802@ospreys.unf.edu

Virtualization Components of the Modern Hypervisor

by

Sean McAdams

A thesis submitted to the
School of Computing
in partial fulfillment of the requirements for the degree of

Master of Science in Software Engineering

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

August, 2015

The opinions and conclusions expressed herein represent the sole opinion of the author and do not necessarily represent the opinion of the University of North Florida or its employees.

The thesis "Virtualization Components of the Modern Hypervisors: A Comparative Evaluation" submitted by Sean McAdams in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering has been

Approved by the thesis committee:

Signature Deleted

Date

12-1-2015

Dr. Roger Eggen
Thesis Advisor and Committee Chairperson

Signature Deleted

12-1-2015

Dr. Sanjay Ahuja

Signature Deleted

12-1-2015

Dr. Behrooz Seyed-Abbassi

Accepted for the School of Computing:

Signature Deleted

12/2/2015

Dr. Sherif Elfayoumy
Director of the School

Accepted for the College of Computing, Engineering, and Construction:

Signature Deleted

12/4/15

Dr. Mark A. Tumeo
Dean of the College

Accepted for the University:

Dr. John Kantner
Dean of the Graduate School

ACKNOWLEDGEMENT

I wish to thank all of the professors who have given me advice and pushed me to complete this thesis after all the time that I have spent on it. I also want to thank my friends who listened to me complain about it constantly while encouraging me to keep moving forward.

CONTENTS

LIST OF FIGURES

ABSTRACT

Virtualization is the foundation on which cloud services build their business. It supports the infrastructure for the largest companies around the globe and is a key component for scaling software for the ever-growing technology industry. If companies decide to use virtualization as part of their infrastructure it is important for them to quickly and reliably have a way to choose a virtualization technology and tweak the performance of that technology to fit their intended usage. Unfortunately, while many papers exist discussing and testing the performance of various virtualization systems, most of these performance tests do not take into account components that can be configured to improve performance for certain scenarios. This study provides a comparison of how three hypervisors (VMWare vSphere, Citrix XenServer, and KVM) perform under different sets of configurations at this point and which system workloads would be ideal for these configurations. This study also provides a means in which to compare different configurations with each other so that implementers of these technologies have a way in which to make informed decisions on which components should be enabled for their current or future systems.

Chapter 1

INTRODUCTION

Virtualization has changed the way the computing industry works. Many different types of virtualization exist, but in the context of this study, I will refer to the type called "hardware virtualization" as just virtualization. Virtualization for purposes of this study will be the emulation or simulation of hardware devices or features using software.

For large companies, such as Microsoft or IBM, virtualization has revolutionized how businesses utilize computer hardware. Still, like most technologies, virtualization products took time to advance to the point where they could be put to widespread use across the industry. Several of these virtualization products have proven to be leaders in the industry and are used heavily in the business world. Across each of these products, we see implementations of very similar if not identical components. Often, different terms or names are used for these components, but each is implemented for a particular purpose such as performance, security, or administration. The focus of this study is to use a subset of these components and perform a quantitative analysis on how each component affects the performance of the systems being virtualized. The majority of the components focused on in this study are the components identified as "paravirtualization" drivers and affect CPU, disk, network, and memory performance. These components are discussed in detail later in the paper. A second focus of this study is to identify a common

and scientific way to analyze the performance effects of these components so that the benefits or detriments of any one component can be quantitatively identified.

Three major virtualization products available today are VMware ESXi (more recently called VMware vSphere), XenServer, and KVM (Kernal-based Virtual Machine). There are other major products on the market, such as Hyper-V, the virtualization technology backed by Microsoft, but this study will focus on the former three. These three were chosen because they are some of the most widely used virtualization products currently on the market and they provide a high level of flexibility to the user when managing specific components [Hwang13].

Each virtualization component has different uses depending on the implementation. Some components (such as the "paravirtualization" drivers) may improve disk I/O, optimize CPU usage, or reduce the RAM used by a virtual machine. However, since any of the major hypervisors can contain dozens to hundreds of components, it is hard to determine which ones help the most and what benefits they give to a particular business. Also, currently, many of the major companies developing hypervisors tend to market towards datacenter-sized businesses. However, small business can benefit from this technology even without having datacenter-scale computing systems to use for virtualization. Reusing old hardware and even using new hardware to its full potential benefits businesses of all sizes. Millions of combinations of hardware and software configurations can be applied to hypervisors, yet there are very few ways of quantitativly comparing these configurations with each other.

"Modern computers are sufficiently powerful to use virtualization to present the illusion of many smaller virtual machines (VMs), each running a separate operating system instance" [Barham08]. This is because of software called the Virtual Machine Manager (VMM) or "hypervisor" on the computer hosting the virtual machines provides emulated hardware to each virtual machine running an operating system. This allows the host to run many virtual machines in parallel, each with its own separate operating system, even if the operating system was originally designed to be an exclusive consumer of the hardware interfaces of a system. The implementation of a hypervisor is very complex.

> "Successful partitioning of a machine to support the concurrent execution of multiple operating systems poses several challenges. Firstly, virtual machines must be isolated from one another: it is not acceptable for the execution of one to adversely affect the performance of another. This is particularly true when virtual machines are owned by mutually untrusting users. Secondly, it is necessary to support a variety of different operating systems to accommodate the heterogeneity of popular applications. Thirdly, the performance overhead introduced by virtualization should be small." [Barham08]

Virtualization technology has been available to the industry since the existence of early mainframes, but only recently has it gained popularity for small business and personal use. Only recently has computer hardware advanced to a point where systems that are capable of running multiple operating systems in parallel are inexpensive and readily available to the public. Under most workloads, a modern operating system does not use a majority of a modern computer system's resources at a given time. Virtualization allows all of a system's resources to be utilized by distributing them over several virtual machines that could be executing different intensity workloads. One of the benefits to

this is that with better resource utilization a company or user can save money by not having to purchase a dedicated physical system for every running operating system.

Chapter 2

ON HYPERVISORS AND VIRTUALIZATION

In order to understand the differences between the major hypervisors currently available, it is important to know that there are two different types of hypervisors. These types are explained below.

2.1 Type 1 Hypervisor

The first type of hypervisor is the "Type 1" hypervisor. This is when the software that drives the hypervisor runs directly on the system hardware. It has a very high efficiency because there is little to no middleware between the hypervisor and the hardware. This is often called a "Bare-Metal" approach. VMWare's vSphere product and Citrix's XenServer are examples of this type of hypervisor. Figure 1 is a diagram of VMWare's hypervisor architecture which is an example of a type 1 hypervisor.

Figure 1: VMWare Architecture - Type 1 Hypervisor on x86 Architecture [VMWare06]

2.2 Type 2 Hypervisor

The second type of hypervisor is the “Type 2” hypervisor; the virtualization layer runs as an application on a host system. The host system handles all of the hardware device support and the virtualization layer handles the emulation of the hardware for the virtual systems. This is considered a “Hosted” architecture. Examples of this architecture can be seen with KVM, VirtualBox, and VMWare Player or Workstation. Figure 2 is an example of a “Type 2” hypervisor, the operating system can run the virtualization layer (hypervisor) alongside other applications, while at the same time the hypervisor can be running its own operating system with applications as well.

Figure 2: Hosted Architecture - Type 2 hypervisor on x86 Architecture [VMWare06]

The differences between the two hypervisor types can sometimes be difficult to determine. In many ways, both types of hypervisors are the same; the main distinction between the two is how the "Host" layer is classified. In a Type 2 hypervisor, it is easy to identify the host operation system because it is being used to interact with the Virtualization layer and additional software can be run in parallel to the virtualization software. While with a Type 1 hypervisor, the only services the hypervisor provides is a very basic management console with no ability to run applications in parallel, it can only be used as a hypervisor. However, even Type 1 systems have a "Host" system, usually

Unix or Linux, in which all the processes required by the hypervisor run. KVM is usually considered a Type 2 hypervisor because it is run on top of Linux; if every Linux module besides the ones required by KVM were removed, then it would essentially be a Type 1 hypervisor. Several distributions of Linux such as Proxmox VE and RHELV (Red Hat Enterprise Linux Virtualization) attempt to do this so that they can get the best performance possible out of a virtualization server.

2.3 Virtualization Types

Instructions executed by the virtual machines have to be translated in some way in order to be executed on the physical resources of the host. When working with modern operating systems there exists four levels of execution privilege, called Rings (levels 0-3). Ring 0 is the most privileged ring and is where an operating system is normally expected to be run so that it has direct control over the physical resources. A hypervisor also expects to be run in Ring 0 and this presents a problem, since the systems it needs to hosts cannot be run in the same ring as the hypervisor. Each hypervisor can potentially host virtual machines in two different ways: Modified or Unmodified. Both of these methods of hosting allow for (and often require) different solutions for how privileged instructions can be executed.

1) Unmodified Guest Systems (Full Virtualization) - When the guest operating system is unmodified and cannot be differentiated from an operating system running directly on hardware. This can be done using several techniques:

a. Binary Translation / Emulation: “This technique is used for the emulation of a processor architecture over another processor architecture. Thus, it allows executing unmodified guest operating systems by emulating one instruction set by another through translation of code [Rodriguez12]. Example Systems are QEMU (which stands for Quick Emulator) and VMWare. QEMU is the package that both KVM and Xen used to provide very basic support for unmodified guest systems. QEMU emulates common hardware devices and exposes them to the guest operation system. Usually the emulated devices are older more commonly supported hardware devices [IBM12A]. QEMU then directs the virtual I/O requests back to the hypervisor where they are translated to real I/O requests (see Figure 3). VMware does a similar process with its Binary Translation. This is useful when a system needs to be virtualized but the guest system cannot be modified.

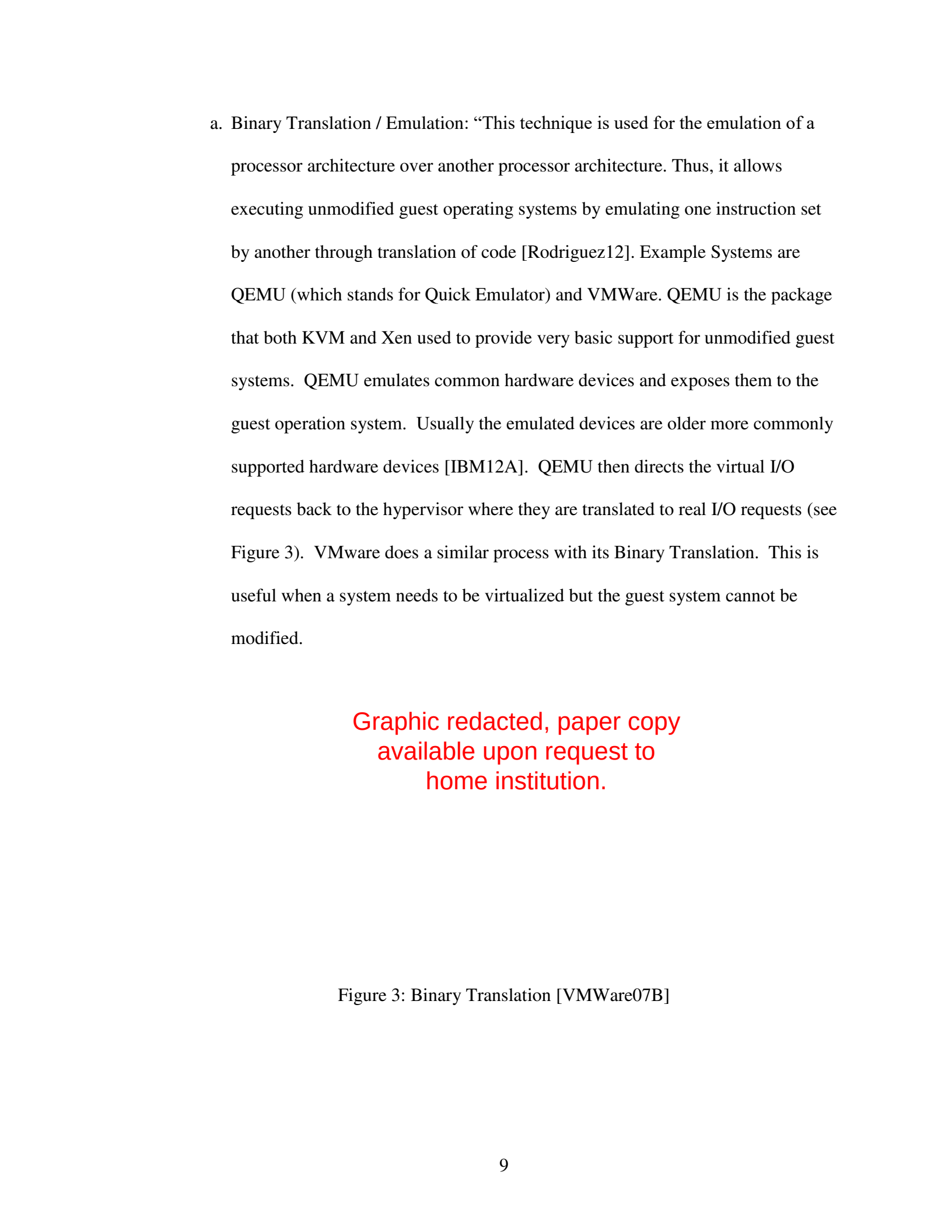


Figure 3: Binary Translation [VMWare07B]

b. Hardware assisted: The hardware itself is designed with virtualization in mind and automatically traps certain system calls from guest systems. This allows some I/O requests to bypass the translation layer and sends them directly to the hypervisor which then determines whether to execute the requests or not. Both Intel and AMD have supported hardware assisted virtualization since at least 2006 [VMWare07B]: Intel's hardware assisted setting is called Virtual Machine Control Structures (VT-x) while AMD's is called Virtual Machine Control Blocks (AMD-V). With these technologies in place "privileged and sensitive calls are set to automatically trap to the hypervisor, removing the need for either binary translation or paravirtualization" (see Figure 4) [VMWare07B].

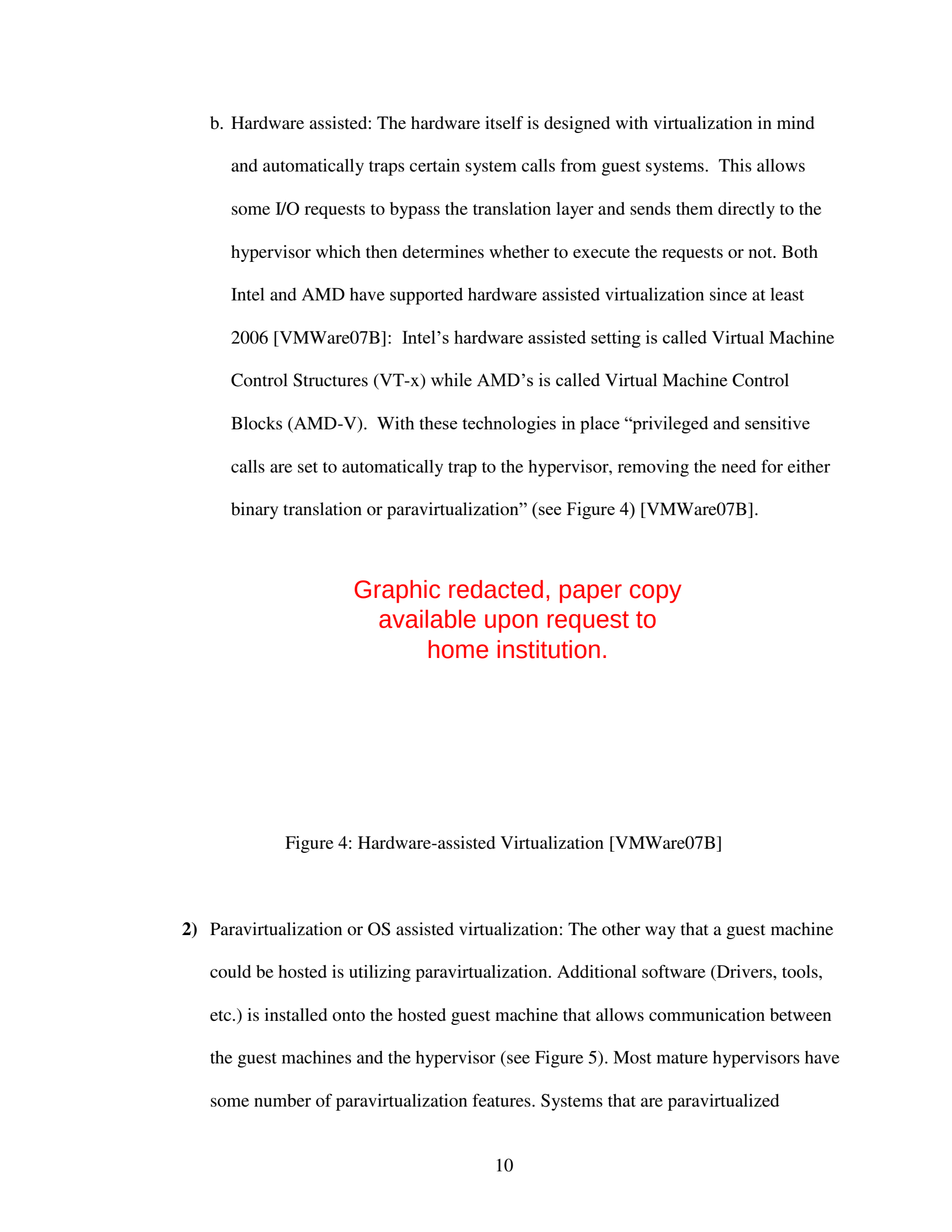


Figure 4: Hardware-assisted Virtualization [VMWare07B]

2) Paravirtualization or OS assisted virtualization: The other way that a guest machine could be hosted is utilizing paravirtualization. Additional software (Drivers, tools, etc.) is installed onto the hosted guest machine that allows communication between the guest machines and the hypervisor (see Figure 5). Most mature hypervisors have some number of paravirtualization features. Systems that are paravirtualized

historically have superior performance to the unmodified guest systems; however, the compatibility of the paravirtualization software and the guest machines tends to be less likely than with emulated solutions [IBM12A]. There are two major disadvantages to the paravirtualization technique:

a. Guest OS must be modified at some point during the virtualization process [Motika11].
b. Every hypervisor tends to create its own version of paravirtualization software, so it is more difficult to move a guest machine from one hypervisor to another [Motika11].

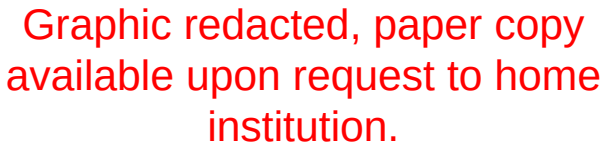


Figure 5: Paravirtualization [VMWare07B]

2.4 Container-Based Virtualization

Another form of virtualization is called "Container-Based Virtualization." This is a parallel concept to a hypervisor but is implemented in a very different way. In this approach, all guest machines need to be the same base images as the host machine. Some

resources are shared such as system libraries and executables. Each virtual machine runs in its own "container" and does not virtualize the hardware of the system since they are all using the same system kernel and the kernel manages the resources of the system (see Figure 6). Systems virtualized in this way sometimes require a custom modified kernel, which needs to be run in all of the virtual machines. Examples of this are OpenVZ, Parallels Virtuozzo Container, and Docker. This approach can significantly reduce the overhead that is seen in hypervisors by removing the redundant kernel level resources [Soltesz07].

This approach will not be tested or explained in extensive detail for this study since it is not a hypervisor and does not follow the same approaches as the other virtualization products in this study.

Figure 6: Container-Based Virtualization [Soltesz07]

Chapter 3

CORE COMPONENTS OF THE MODERN HYPERVISOR

As stated before, each hypervisor creates its own implementation of components, such as the paravirtualization tools, which are intended to improve some aspect of the virtual system. It is difficult and less productive to breakdown each and every component that each and every hypervisor implements so only a subset of components will be discussed. Also, while this study focuses on the use of the Microsoft Windows Server operating system as the guest systems for the actual experiment, all of the core components below are also available to many Linux distributions and other operating system. Of the components below the ones marked as "WT" (Will Test) will be analyzed for performance benefits in this study. "Will Test" compnents were chosen based on performance suggestions in the reference material ([VMWare11], [RedHat14], and [IBM12A]). The other components are included to have a more complete list of available components for hypervisors in general and as potential future work based on this study.

There are two categories of components available when using a hypervisor:

1) Components that when configured change how the guest machine acts, either by changing the hardware the guest can see and/or access or by modifying the guest directly.
2) Components that are enabled on the hypervisor, in which the guest machines are unaware of the modification.

Components that affect the way guest machines act by modifying the guest directly are considered paravirtualization components. Each hypervisor implements these slightly differently. VMWare's implementation is included as part of its VMware Tools installation. XenServer paravirtualization drivers are included in its software package called XenServer Tools, and KVM has a set of windows drivers called VirtIO-win that are provided for free by Fedora. Because of the disparity between the implementation of the VMWare and XenServer tools these drivers are not compatible with any other hypervisor other than the one they were designed for. The VirtIO drivers, however, are based on a standard that was presented to unify the different virtual I/O implementations currently present on Linux [Russell08]. It is the hope of the designers and developers of the VirtIO interfaces that VMWare and Xen will adopt the same standards, which would allow the guest machines to easily be moved from one hypervisor to another [Russell08]. The other components that change how the guest machine act are the interfaces in which the emulation device exposes to the guest operation system. QEMU has a wide variety of devices that it supports emulation for, some that are intended to provide improved performance and some that provide a wider range of operating system compatibility. Determining which of these is best suited for the virtualization environment being used can be time consuming and difficult.

Common components in the first category include: Paravirtual Block (Disk) Drivers, Paravirtual Network Drivers, Paravirtual Memory Drivers, Huge Pages/LargePage, Network Bridging, Network Address Translation, Memory Compressions, Hypervisor

Disk Swapping, Resource Overcommiting, Kernal Samepage Merging/Transparent Page Merging, and SSD Support.

3.1 Paravirtual Block (Disk) Drivers (WT)

KVM, by default, uses QEMU to emulate an IDE disk type. When Microsoft Windows Server 2008 R2 is running the system sees it as "QEMU HARDDISK ATA Device." It is supported by many versions of the Microsoft Windows operating system. For the paravirtual disk support KVM uses the VirtIO drivers. When loaded into a Microsoft Windows Server 2008 R2 system it displays "Red Hat VirtIO SCSI Disk Device" as the driver name and as the name suggests it is a SCSI type device. In order for the VirtIO drives to be installed the Operation System must be pre-loaded with the driver before installation or the system needs to be started with another disk interface type (such as the default IDE interface) so that the paravirtualization drivers can be installed and then the system can be modified to use the VirtIO disk type.

VMWare is different from both XenServer and KVM in that its default disk type is not an IDE interface, it is instead a SCSI interface. The difference between the default SCSI interface and the paravirtual SCSI interface is the storage controller installed with VMWare tools. For both the default and the paravirtual configurations the driver displayed by the operating system shows "VMware Virtual disk SCSI Disk Device." For the default configuration, however, the storage controller shown was "LSI Adapter, SAS 3000 Series" while the paravirtual configuration displayed "Vmware PVSCSI

Controller." The paravirtualization drivers are included with VMware Tools, which can be installed into the operating system after it has been installed onto the guest machine. "VMware Paravirtualized SCSI (PVSCSI) is a special purpose drive for high-performance storage adapters that offer greater throughput and lower CPU utilization for virtual machines. They are best suited for environments which guest applications are very I/O intensive [VMWare09B]." It is important to note that the PVSCSI driver is not supported by all operating systems as it is designed for newer systems, Microsoft Windows Server 2008 R2 is one of the supported systems.

XenServer, by default, uses the same emulated IDE disk that KVM does (by using QEMU). However, its paravirtual disk drivers have less recent documentation available for them. When loaded the Windows guest system sees the paravirtual driver as "XENSRC PVDISK SCSI Disk Device." According to the official Xen webpage "PVSCSI allows high performance passthrough of SCSI devices (or LUNs) from dom0 to a Xen PV or HVM guest. PVSCSI can be used to passthrough a tape drive, tape autoloader or basically any SCSI/FC device. By using PVSCSI the guest can have direct access to the SCSI device (required by for example some management tools). PVSCSI can also be used to passthrough multiple SCSI devices or the whole SCSI HBA" [Xen15].

3.2 Paravirtual Network Drivers (WT)

Each of the products in this study provide a way of using the e1000 Network Adapter, a gigabit network adapter that is widely supported by operating systems. However, by default, XenServer does not expose this driver as its default emulated network device. For this study, this was remedied by replacing the default emulated device (rtl8139) with the e1000 device. This allows the network configuration across the three hypervisors to be as close as possible, providing a better performance comparison. When loaded onto a Windows guest, the e1000 adapter appears as "Intel(R) PRO/1000 MT Network Connection."

With KVM, the network paravirtual driver for Windows guests is called virtio-net and appears as "Red Hat VirtIO Ethernet Adapter" when installed. With the VirtIO drivers "when the guest OS performs a network instruction, the instruction is handled by the virtio kernel module on the guest OS. The guest machine does not try to perform OUT instruction (as in the case of emulation) therefore, the processor is not moved to the root operation mode, and there is no VM exit (the VM exit is called when the queue is out of buffers, consequently the instruction cannot be handled)" (see figure 7) [Motika11].

Graphic redacted, paper copy available upon request to home institution.

Figure 7: KVM Emulated Network Device (left) vs. VirtIO Paravirtual Network Device (right) [Motika11]

VMware's paravirtual network device is called VMXNET3 and appears as "vmxnet3 Ethernet Adapter" in the guest operating system. "The paravirtualized network adapters in the VMXNET family implement an idealized network interface that passes network traffic between the virtual machine and the physical network interface cards with minimal overhead" [VMWare11]. VMXNET3 is the third generation of VMware's network paravirtualization drivers and supports many new features that previous versions did not. It was introduced in VMware ESXi 4.0 and was a complete rework of the previous generation (VMXNET2) [VMWare09A]. VMXNET3 is available through the VMware Tools installation and is installed at the same time as the paravirtual disk drivers. XenServer's paravirtual network device is called "Citrix PV Network Adapter #0", it is included with the install of XenServer tools. "Privileged domains, called `driver' domains, use their native device drivers to access I/O devices directly, and perform I/O operations on behalf of other unprivileged domains, called guest domains. Guest domains

use virtual I/O devices controlled by paravirtualized drivers to request the driver domain for device access" (see Figure 8) [Cox06].

Figure 8: Xen paravirtual network architecture [Cox06]

3.3 Paravirtual Memory Drivers (WT)

For all three hypervisors the main purpose of the memory drivers is to provide memory mallooning. In a virtual environment, often some virtual machines will be idle more often than others. While those machines are idle their memory is being unused. With memory ballooning the hypervisor can reclaim some of that memory for use in other systems that need it more. This is only required when resources are over committed (discussed in more detail below) and guests machines are allocated more cumulative memory than is available to the physical machine. KVM's driver for memory ballooning

is the VirtIO Balloon (virtio-balloon) driver. When creating a virtual machine in a KVM environment the minimum and the maximum allocated virtual memory can be defined and in the machine definition. VMware's driver is called "vmmemctrl" and the minimum and maximum memory allocations can be set when creating the virtual machine through the creation wizard. In XenServer this processes is part of their "Dynamic Memory Control" and can be configured using their XenCenter management interface or through direct modification of the machine definition.

3.4 Huge Pages/LargePages

Large Pages allow the hypervisor to allocate memory pages in 2MB sets instead of 4KB sets. This can reduce Translation Lookaside Buffer (TLB) misses since larger memory ranges can be stored in the TLB. It is supported by all three hypervisors but requires specific configurations of the host and guest systems. KVM has two different version, the first is the standard "Huge Pages" which is actually enabled on the Linux host itself, the second is "Transparent Huge Pages" which can be enabled for certain guest machines [RedHat14]. In XenServer "Huge Pages" can be enabled in the Linux host in a similar way to KVM. In VMware the setup is done automatically when the guest OS is set to use Large Pages. For Microsoft Windows based systems Large Pages must be enabled through the OS for it to take advantage of the change on the hypervisor.

3.5 Hypervisor Enabled Components

3.5.1 Network Bridging vs. Network Address Translation (NAT)

Network bridging is where the hypervisor simulates a physical network connection for the guest. The guest is given a physical IP address and can be access directly on the LAN/VLAN that it is part of. In Network Address Translation (NAT) all traffic is routed through the hypervisor's physical network interface and the guest machine is only accessible through the hypervisor.

3.5.2 Memory Compression

In memory compression the hypervisor compresses memory pages in a compression cache during high memory utilization. This reduces overall memory usage but can also reduce overall memory performance since it will need to decompress the pages before using them again [VMWare11].

3.5.3 Hypervisor Disk Swapping

Operating systems have used swapping for managing high memory usage for a while; however, allowing the hypervisor to swap memory for the guest machines increases performance compared to classic operating system based swapping techniques. The

hypervisors can better manage the location that the memory is swapped to because it has access to the entire physical disk while the guest machine has access to only the space allocated to it by the hypervisor [VMWare11].

3.5.4 Resource Overcommitting

In this process the host machine's physical resources are over allocated to multiple guest machines. An example of this over allocation would be allocating two guest machines 4GB of virtual memory each on a host that has only 6GB of physical memory available. This allows a user to continue to run guest machines even if the host has reached capacity for certain resources. This is possible because the host manages the resources and re-allocated unused resource to other guest machines. However, "over-committing processor resources can cause performance penalties. When a processor must switch context from one process to another process, the switch affect performance of the system" [IBM12A].

3.5.5 Kernel Samepage Merging / Transparent Page Sharing

Often related to memory over-commitment, page sharing is a useful technique for saving memory for a hypervisor. This process allows the hypervisor to share duplicate pages across guest machines by identifying duplicate pages in memory and merging them without the guest machine's knowledge. However, besides the performance penalties that it incurs, there is also a limitation to this technique; in that many hypervisors (such as VMWare) do not allow sharing of large pages [VMWare11].

3.5.6 SSD Support

With the advent of Solid State Disk (SSD) technologies many of the current disk performance bottle necks no longer exist. Many hypervisors have steps a user can take to optimize performance for systems with an SSD, such as using an SSD as a swap partition [VMWare11].

Chapter 4

A LOOK AT THE HYPERVISORS OF TODAY

There are many hypervisors in today's market, it would be impractical to list and describe all of them in a single document. However, some hypervisor implementations stand out from the rest as industry leaders [Hwang13]. These hypervisors include:

4.1 VMWare VSphere/Workstation/Player

One of the largest providers of virtualization products today is VMWare. They provide enterprise and personal virtualization solutions for a variety of system configurations. The one used in this study was VMWare VSphere, also known as VMWare ESXi, and is marketed as a business solution. It is intended to be installed as the operating system on a physical system. However, many of the other solutions provided by VMWare run as applications in an existing operating system. These solutions are VMWare Player, VMWare Workstation, and VMWare vFusion. All of these are considered Type 2 or "hosted" hypervisors and run in parallel to an existing host operating system. VMWare Player is a free (for personal use) product that has very basic capabilities for hosting virtual machines. VMWare workstation is a pay to use product with more extensive features, such as snapshots and many of the advanced configuration settings. VMWare vFusion is a Mac OSX based solution to allow virtualization on most Apple OSX based systems and provides functionality comparable to VMWare workstation.

VMWare vSphere, however, is the flagship Type 1 or "bare metal" hypervisor solution provided by VMWare. This has all of the features of the hosted products plus an extensive list of custom configurations and even includes remote and local command line management interfaces. VMWare VSphere is intended as an enterprise business product and can be scaled to support very large infrastructures. However, even though it is marketed as an enterprise application it also functions very well for small scale systems, such as in a small business environment or in a small software team.

4.2 Xen/XenServer

Xen is an open-source hypervisor that was first introduced in 2003 with the publishing of "Xen and the Art of Virtualization" [Barham08]. It was originally developed with idea that, while "allowing unmodified operating systems to be hosted" is extremely useful, "It also has its drawbacks" [Barham08]. It was built with paravirtualization in mind, where the guest operating system can be modified before being hosted on the hypervisor. Currently Citrix is major supporter of Xen, their enterprise operating system called XenServer provides a bare metal implementation of Xen and automatically provides many of the changes required for a guest machine. The source code for Xen is Open Source and available online.

4.3 KVM

KVM stands for Kernel-based virtual machine, it is an open-source hypervisor that is included as part of the Linux kernel as of version 2.6.20 which came out in February of 2007 [Linux07]. At first glance it appears that it would be a Type 2 hypervisor, but when all un-required modules of the Linux kernel and the software installed by default are removed it is essentially a Type 1 hypervisor. KVM was originally created by a company called Qumranet in 2006. At the time Xen was the most widely used open-source hypervisor, but it required a custom built operating system to utilize. KVM was created as a module in the Linux kernel which allowed for fast adoption and modifications [IBM12B].

"The fact that the Virtual Machine Monitor (VMM) is part of the kernel model reduces the cost of switching between the VMM mode and the host mode. This also makes the KVM thinner than other VMMs, because the functionality of running a guest OS is not needed on the KVM, and neither is the memory management. The guest OS is a regular Linux process: it is scheduled like other processes, and its memory is managed by the Linux kernel" [Motika11].

4.4 Hyper-V

Hyper-V is Microsoft's hypervisor solution for the Microsoft Windows operating system. It was released well after VMware, Xen, and KVM but it focused on virtualizing systems

using the Windows kernel as a base rather than the Linux kernel. Hyper-V has been available since Microsoft Windows Server 2008 and a slimmed down version of the technology, Microsoft Hyper-V Server, was released shortly after that removed the overhead of the host Windows installation and simulated a Type 1 hypervisor.

4.5 Oracle VM Server/VirtualBox

Another contender for hypervisors in the software industry is Oracle and their Oracle VM Server and Oracle VM VirtualBox. The Server solution is intended for datacenters and would be considered a Type 1 hypervisor while their VirtualBox solution is a Type 2 product and is available as an application for many operation systems. [Kumar10]

4.6 Parallels Desktop/Virtuozzo Containers

Another set of virtualization solutions that has been gaining momentum in the virtualization industry are the products created by the company Parallels, Inc. Their Parallels Desktop product is the most widely known solution allowing virtualization of several different operating systems on the OSX operating system for Macintosh computers. This is considered a Type 2 hypervisor as it is an application. The Parallels Virtuozzo Containers product is a container-based virtualization solution targeted towards the datacenter and enterprise markets.

Chapter 5

PURPOSE AND GOALS

The goal of the study is to identify common hypervisor components that could be used to increase system performance, evaluate how each of them affects the performance of a virtual machine, and provide a method in which to compare these components across hypervisors. As an example, paravirtual network drivers are often used to provide additional network performance, but there is little information on how significant that increase is. In addition, all of the industry leading hypervisors have some implementation of paravirtualization drivers; however, very little information is available showing the performance difference of these drivers across hypervisor products.

Most performance comparisons such as [VMWare07A], [Suganaya12], [Che08], [Mohan12], [Deshane08], and [Chierici10] may implement some of these components (though most did not mention that they even knew they existed) but do not look at the performance improvements from each components, only of the entire hypervisor. When creating a real world virtual environment an out-of-the-box configuration is rarely left unmodified, time is taken to tweak performance based on how that system is planning to be used. It would be beneficial to know what components to enable and what performance improvements can be gained from implementing these components when a system is initially setup or while tuning performance over time for particular workflows. This study is intended to identify some of these components, explain what they do, and

determine what performance benefits they give, and provide a way to quantitatively compare their performance differences. As virtualization products evolve and other hypervisors appear, these virtualization products will need to implement these or similar sets of components to be competitive and users will need ways of evaluating these features for use in a production system.

In this study, three of the most popular and powerful virtualization technologies currently available are used: VMware vSphere (also known as VMware ESXi), XenServer, and KVM. With these three technologies, a defined set of components are implemented and performance tests are performed on a Windows Server 2008 R2 virtual machine to gauge what benefits the components provide. Microsoft's Windows Server 2008 R2 was the chosen operating system because it is very commonly used in enterprise environments. This topic would likely be beneficial to anyone implementing virtualization technology on a small or large scale. However, this would be more beneficial on a small scale since the hardware being used is closer to small business infrastructure hardware than the infrastructure of companies that can afford their own datacenters and/or dedicated servers. Large companies would likely invest a lot of money and time to optimize their virtualization systems; however, this paper would be a highly beneficial start to that level of research. What this study does provide for both large and small companies is a method with which to test these virtualization components individually, which is currently not available.

Chapter 6

EXISTING RESEARCH

Several studies provide performance comparisons between sets of hypervisors. However, no study focuses on the individual components of a hypervisors. There are also many studies describing how to implement individual components of specific hypervisors, which tend to be mostly white papers or technical manuals written by their respective companies. Still, none of those studies provides performance benchmarks on these components.

The study that was most similar to this research is [Suganaya12], written by a student at the University of North Florida. The researcher did a performance comparison with the SPECvirt_sc2010 benchmark of the three hypervisors that being used in this research study. The difference between this study and the study of [Suganaya12] is that there is no documentation provided stating that additional components or features were enabled on the hypervisor. Only standard installs of all three hypervisors appear to be used in [Suganaya12]. A problem with using the default installations is that VMWare and XenServer have several of the components enabled by default while KVM does not, this is at least one cause of the performance disparities seen in that study. Another difference between [Suganaya12] and this study is that SPECvirt_sc2010 is not used, instead PassMark Performance Test is used. The SPEC benchmark is focused on testing multiple workloads running on multiple virtual machines and uses older operating system

workflows. PassMark has a series of benchmarks geared towards testing individual machine performance for specific resource types. Therefore, the PassMark software will act as a normal application and have no knowledge of other systems running on the hypervisor. Using PassMark, this study will show how the systems perform under parallel homogenous workloads and how that performance might degrade as system counts increase.

Another very similar study to [Suganaya12] is [Xianghua08]. It compares KVM, Xen, and VMWare using a series of tools: UBench, IOZone, Netperf, Sysbench, and a couple custom benchmarks for Gzip and LAME performance. In [Xianghua08] the authors do not explore enabling any of the performance components even though they do mention it as possible future work. In addition, their base systems for Xen and KVM were on CentOS while Fedora will be used in this study, which is constantly updated with the latest versions of KVM, since it is maintained directly by the Red Hat Corporation.

Another research project with objectives similar to those of this study is [VMWare07A]. It is a performance study done by VMware using a couple testing tools including PassMark (one of the testing tool used in this study). The authors use the CPU and memory benchmarks from PassMark to compare a system hosted native (non-virtualized) to systems hosted on ESX and Xen. In almost all the tests they performed Xen performed worse than VMWare ESX. The tests were done on Windows Server 2003 which was a very common operating system then, however, Windows Server 2008 R2

will be use in this study since many corporations have migrated to the later versions of Windows Server.

Similar to [VMWare07A], [Che08] presents a similar project where a performance comparison between two hypervisors was performed, however, it focuses on Xen and KVM for its performance testing. It also uses a different set of benchmarks, LINMARK (used for the CPU benchmark), LMBench (used for the memory benchmark), and IOZone (for I/O benchmarks). Their tests were performed on Xen and KVM hosted Fedora Core 8 and utilized Windows XP virtual machines. In [Che08] a majority of the performance tests showed Xen with better performance results. It is worth noting that these systems (Xen/KVM) have been through many changes since 2008 and the results in this paper are outdated. However, it is a good reference for comparing how these systems have improved in the last few years.

[Chierici10] and [Deshane08] also focus on comparing the Xen and KVM hypervisors. [Chierici10] focuses on general performance, performance isolation, and scalability. The authors used IOZone and the Phoronix Test Suite for their general performance tests and SPECweb2005 for their performance isolation tests. To test scalability, they compiled Apache on a single guest and then slowly increased the machine count and ran the same command in parallel on all the machines. [Deshane08] used three performance tools: hep-spec06 (for CPU performance), iperf (for network performance), and bonnie++ (for disk performance). Again, both studies found Xen to have the best performance and stability.

Chapter 7

METHODS AND RESULTS

7.1 Methods

Four system images were created on each hypervisor using the configurations described in the Machine Definition section below. Two were setup with the no modifications configuration and two were setup with the paravitualization configuration. Tests were performed with the following permutations on each hypervisor:

For Disk, CPU, and Memory tests:

1) Single no modification guest
2) Two no modification guests in parallel
3) Single paravirtual configuration guest
4) Two paravirtual configuration guests in parallel

Disk CPU and Memory tests were performed using PassMark PerformanceTest 8.0.

For network tests:

1) Two no modification guests, one as server, one as client
2) Two paravirtual configuration guests, one as server, one as client
3) Two no modification guests using the e1000 Ethernet emulator, one as server, one as client

Network tests were performed using NT TCP Testing Tool.

Every test was run a minimum of 30 times (general considered to be a good number for student t-distribution tests [Hogg10]) per machine in order to get a reliable set of measures. 9 CPU tests, 3 disk tests, 7 memory tests, and 1 network test (a total of 20 tests) were performed on each of the guest machines. For the single machine tests (1 and 3 above) were triggered from a remote machine using the psexec.exe utility and a batch file (included as Appendix G). For the multiple machine tests the psexec.exe utility was used to execute the ParallelExecution.exe tool (source code included as Appendix H) which orchestrated the synchronization of the tests across each machine. Each test was run 30 times in sequence using the scripting functionality of the PassMark PerformanceTest tool by creating a script file configured to run a specific test 30 times (see Appendix G). Results for each test were written to a comma-separated values (CSV) file as they completed. Results for the all the tests were collected using the batch files that executed the tests via the xcopy utility. Tests were performed in this manner to eliminate variability and ensure all tests were executed identically.

The network tests were also executed remotely using the psexec.exe tool. The NT Testing TCP Tool requires a client and a server for testing, all tests were performed using one guest machine as the server and another as the client. The results of the test were written to a text file and collected with the xcopy utility. Additional batch files were created for this process as well to ensure consistency (see Appendix G).

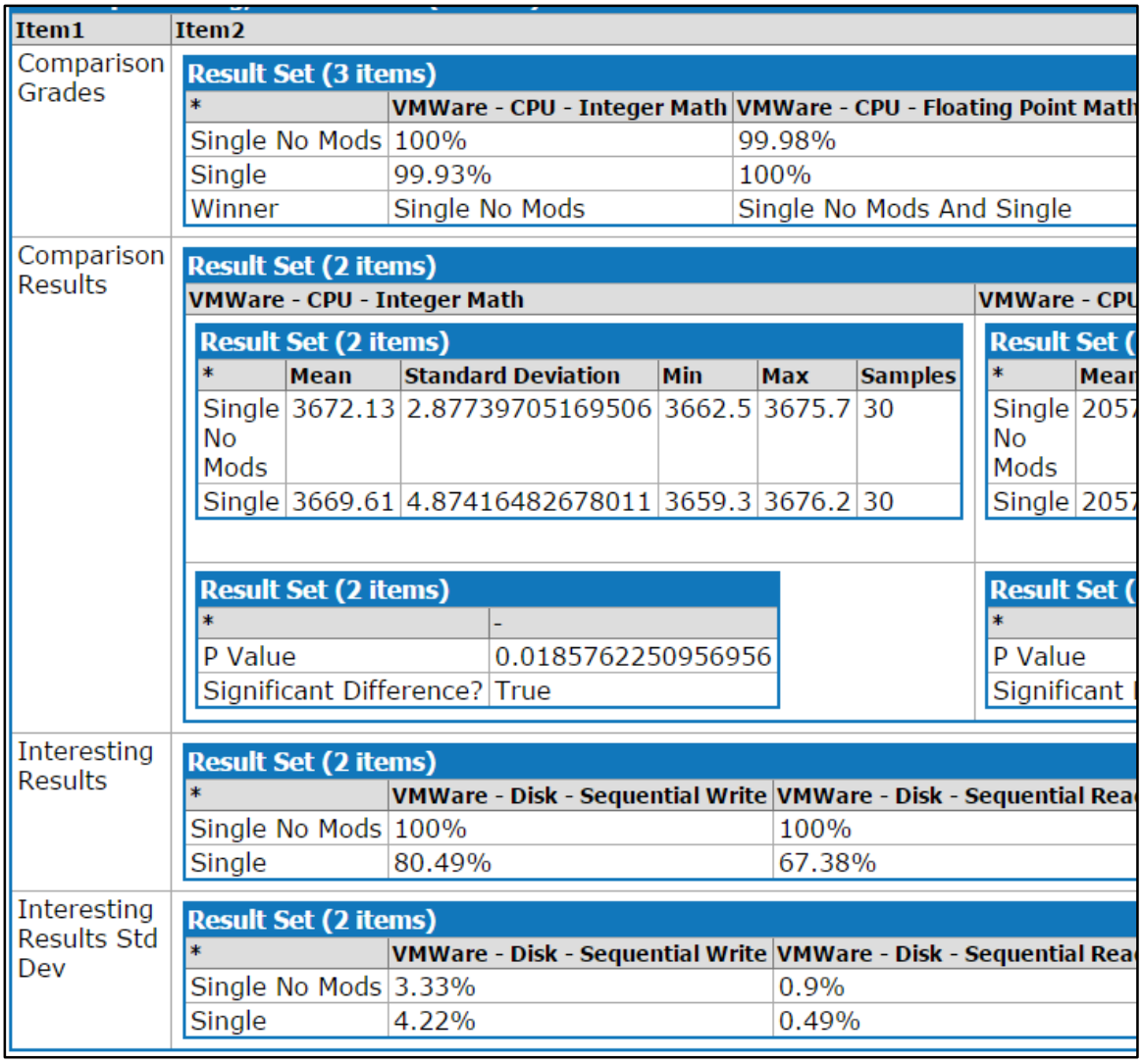
Data analysis and aggregation were done in C# using LinqPad as the development tool. LinqPad was chosen for its light footprint and ability to create C# scripts quickly. Two statistical analysis packages were used to run the tests for each data set, multiple packages were used to ensure correctness for the test results. The first statistical package used was R, a programming language with libraries specializing in statistical analysis. Interaction with R was facilitated by R.NET, an open source library that allows for easy communication between Microsoft .NET based applications and the R programming language. The second statistical analysis package used was the TDistribution class from the Microsoft .NET library (System.Web.UI.DataVisualization.Charting.StatisticFormula). The StudentDistribution methods used were isolated and pulled out into the LinqPad script so that they could be more accessible (see Appendix F).

The output files from the test runs are parsed programmatically with the LinqPad script in order to pull the data into a more manageable format. The different result sets were then compared as follows:

1) Parallel no modifications(PNM) vs. Parallel with paravirtualization(PWP)
2) Single no modifications(SNM) vs. Single with paravirtualization(SWP)
3) Single no modifications(SNM) vs. Parallel no modifications(PNM)
4) Single with paravirtualization(SWP) vs. Parallel with paravirtualization(PWP)
5) Cross hypervisor comparison for single no modifications(SNM)
6) Cross hypervisor comparison for single with paravirtualization(SWP)
7) Cross hypervisor comparison for parallel no modifications(PNM)

8) Cross hypervisor comparison for parallel with paravirtualization(PWP)
9) Comparison of all three network configurations: paravirtualization vs. no modifications vs. e1000 network adapter.
10) Cross hypervisor comparison of network performance for both single and parallel test configurations

Comparison results were then displayed in several ways in order to extract useful comparisons (see Figure 9). The first way was a grade based on the highest mean for the data sets, labeld “Comparison Grades” in Figure 9. The set with the highest mean is given a grade of 100% and the set(s) with the lower mean is given a grade relative to the set with the highest mean (the “Winner” in Figure 9). The second way was an aggregates view which displayed a comparison of the mean, standard deviation, the smallest value, the largest value, and the number of items in the data set for each pair of data sets compared (see “Comparison Results” section of Figure 9). Immediately below the display of the aggregate data in the same section is the p-value from a TTest between the data sets and whether that p-value indicates that the sets are statistically different. The sections labeled “Interesting Results” and “Interesting Results Std Dev” are used for creating the result graphs in the next section. Each data set was normalized using a calculated interquartile range (IQR) with a generous margin to prevent it from removing important results. The IQR is the range in which a majority of the items in a data set fall into, this removed any extreme outliers from the data set that may have been caused by unexpected circumstances (see Appendix F for implementation).



Item1	Item2
Comparison Grades	Result Set (3 items)

*	VMWare - CPU - Integer Math	VMWare - CPU - Floating Point Math
Single No Mods	100%	99.98%
Single	99.93%	100%
Winner	Single No Mods	Single No Mods And Single

Comparison Results	Result Set (2 items)

VMWare - CPU - Integer Math

Result Set (2 items)

*	Mean	Standard Deviation	Min	Max	Samples
Single No Mods	3672.13	2.87739705169506	3662.5	3675.7	30
Single	3669.61	4.87416482678011	3659.3	3676.2	30

Result Set (2 items)

*	-
P Value	0.0185762250956956
Significant Difference?	True

VMWare - CPU

Result Set (

*	Mean
Single No Mods	2057
Single	2057

Result Set (

*
P Value
Significant

Interesting Results	Result Set (2 items)

*	VMWare - Disk - Sequential Write	VMWare - Disk - Sequential Rea
Single No Mods	100%	100%
Single	80.49%	67.38%

Interesting Results Std Dev	Result Set (2 items)

*	VMWare - Disk - Sequential Write	VMWare - Disk - Sequential Rea
Single No Mods	3.33%	0.9%
Single	4.22%	0.49%

Figure 9: Section of output for the statistical comparison from the LinqPad script

7.2 Results

Several of the result sets ended showing statistically insignificant differences or statistically significant differences but by a very slim margin. This section will focus on the "Interesting Results", defined as data points in the comparison set that vary by more than 10% ***and*** the difference is statistically significant (p-value < .05 using the null

hypothesis). This means that ***all results not displayed showed an insignificant amount of difference*** or no change. For example, in the comparisons across each hypervisor types the tests not displayed are the tests that showed no interesting difference between each hypervisor.

The graphs in the next sections are built using the "Interesting Results" and the "Interesting Results Standard Deviation" sections of the LinqPad script results (see Figure 9). The Y-Axis of the graphs (Figures 10-19) represent the "Interesting Results" of all the "Comparison Grades" as described earlier. The error bars represent the standard deviation percentage relative to the highest mean for that set of results (the "Interesting Results Std Dev" section of Figure 9).

7.2.1 Parallel no modifications (PNM) vs. Parallel with paravirtualization (PWP)

The most surprising outcome of the tests for this category was the VMWare tests. There were minimal changes in performance with the introduction of paravirtualization, only 3 (2 shown in Figure 10) of the 19 tests showed any improved performance. This shows that the default installation of VMWare is well tuned. In comparison the XenServer tests showed an increase in performance for 14 (4 shown in Figure 10) of the 19 different tests and KVM showed improvement in 13 (6 shown in Figure 10) of the 19 tests.

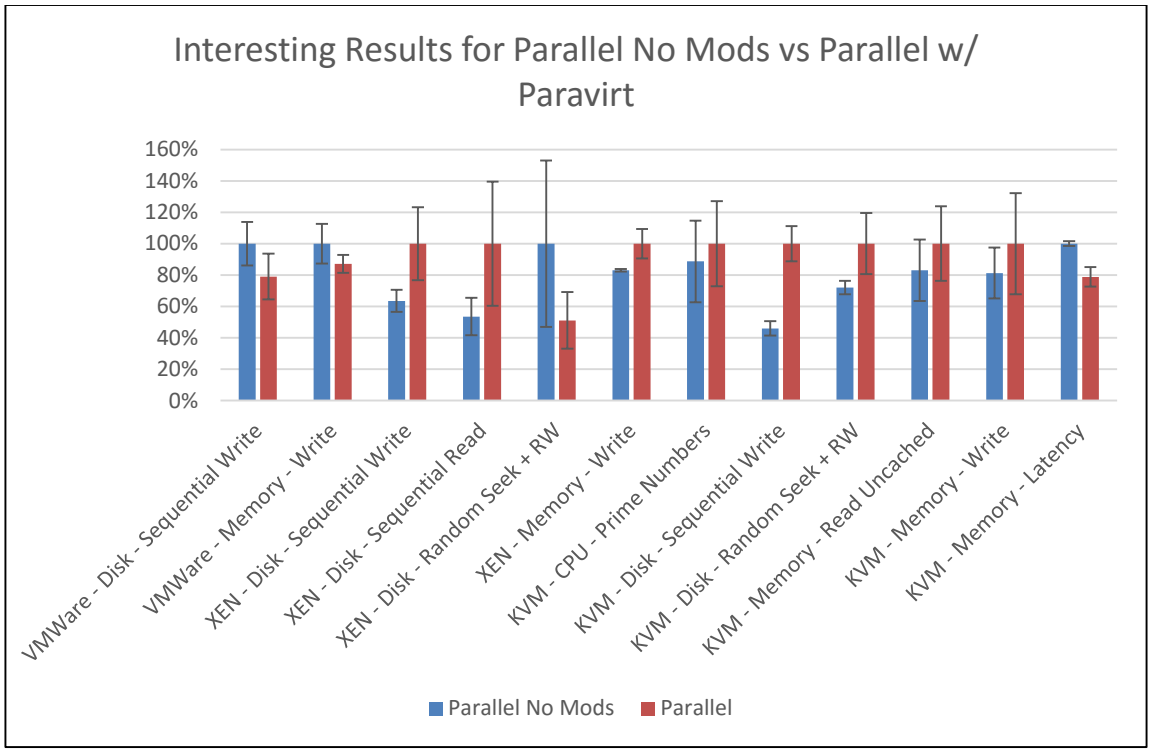


Figure 10: Results for Parallel No Modifications vs. Parallel with paravirtualization

7.2.2 Single no modifications (SNM) vs. Single with paravirtualization (SWP)

The results for the single machine tests were very similar to the results for the parallel machine tests. However, the most interesting set of results from this category are from "Random Seek + RW" in the XenServer's SWP configuration. The drastic performance degradation is surprising from a system built with Xen, the hypervisor which used to rely on paravirtualization for its performance. Overall VMWare saw very little performance improvements for 7 of 19 tests and a performance degradation for 3 of 19 tests when using the SWP configuration. XenServer saw an improvement for 10 (2 shown in Figure 11) of 19 tests and a significant degradation on 1 of the 19 tests. KVM saw improvement for 11 (6 shown in Figure 11) of 19 tests.

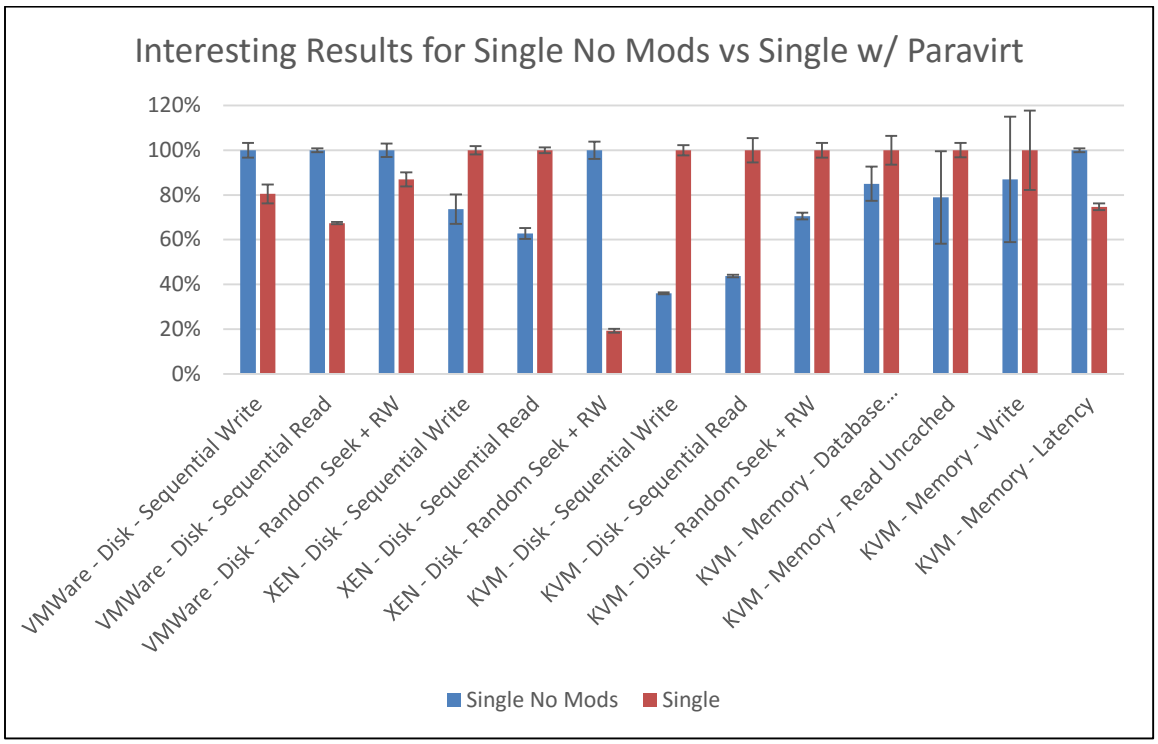


Figure 11: Results for Single no Modifications vs. Single with paravirtualization

7.2.3 Single no modifications (SNM) vs. Parallel no modifications (PNM)

As expected with a comparison of the SNM and PNM configurations the SNM configurations consistently had the best performance. With any system where multiple processes are running simultaneously, one or both of the processes will suffer to some extent. The main focus of this comparison is to see which of the three systems had the least degradation from the SNM configuration to the PNM configuration. Many of the CPU based tests have very little change between the two configurations. This is likely because each machine was given a dedicated core on the host machine so there was minimal contention for processor resources. Both XenServer and VMWare showed a

decrease in performance from SNM to PNM in 9 of 19 tests. KVM showed a decrease in performance for 7 out of 19 tests (see Figure 12). The most interesting degradations in this test category are the disk performance tests, in which every hypervisor saw a decrease of more than 50%. This decrease means that with parallel machines running the overall disk performance is reduced even though multiple machines are running at the same time. An important thing to consider in this category is that any performance reduction of more than 50% means that the overall system performance decreased and thrashing occurred. This is because the average of these tests (the values displayed in the graphs) are taken from individual machines which means that for the parallel machine configuration all results would be doubled to get total system performance of the host for a given test.

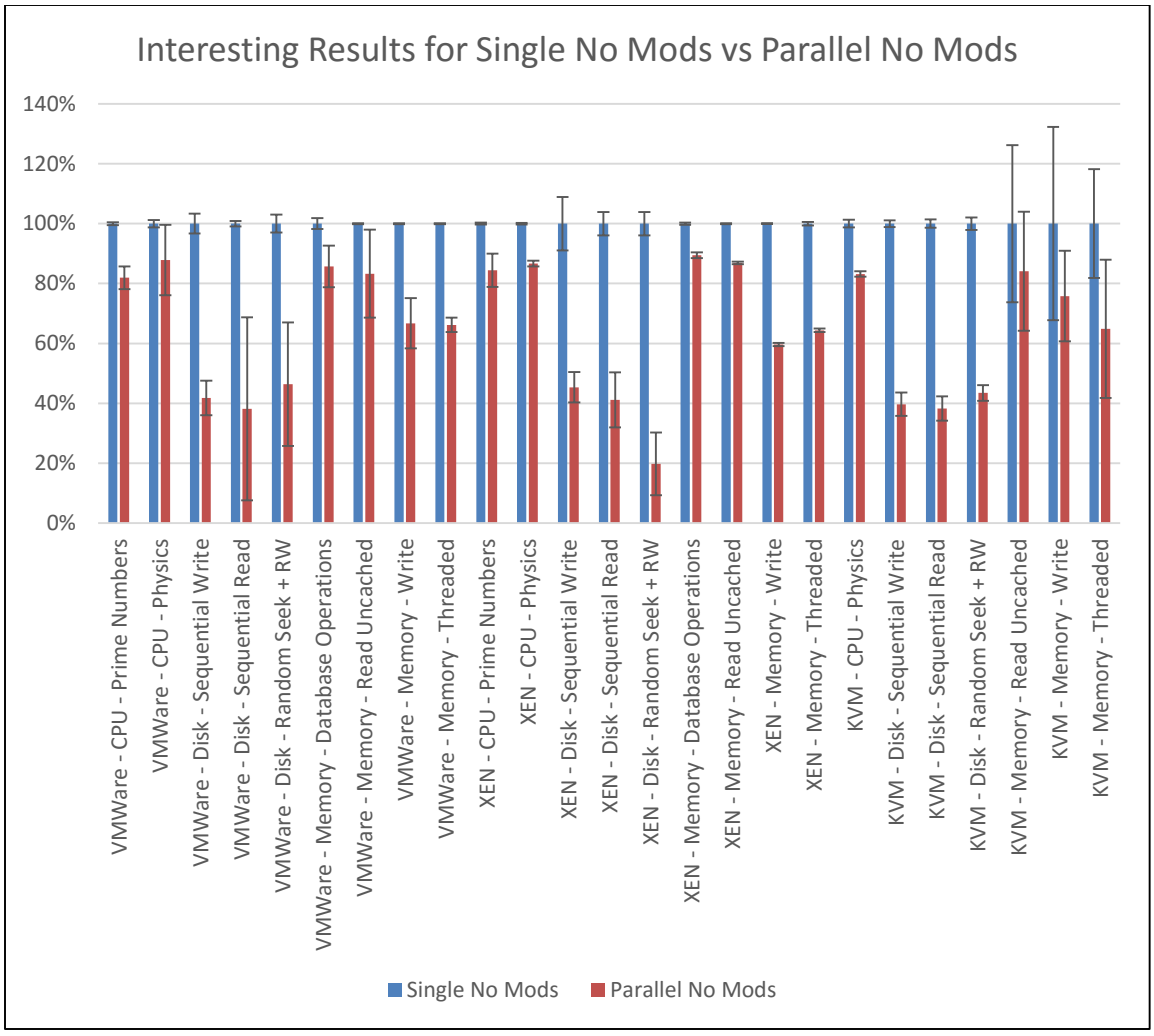


Figure 12: Results for Single no Modifications vs. Parallel no Modifications

7.2.4 Single with paravirtualization (SWP) vs. Parallel with paravirtualization (PWP)

Similar to the SNM vs. PNM results all of the interesting results in the SWP vs. PWP tests show the SWP configuration to have the better performance. The major differences of this test comparison to the SNM vs. PNM test is that 2 out of 3 of the VMware disk tests and all of the Xen disk tests now had a less than 50% performance reduction.

Meaning that overall disk performance increased for the host. KVM still showed an overall decrease in disk performance. (see Figure 13)

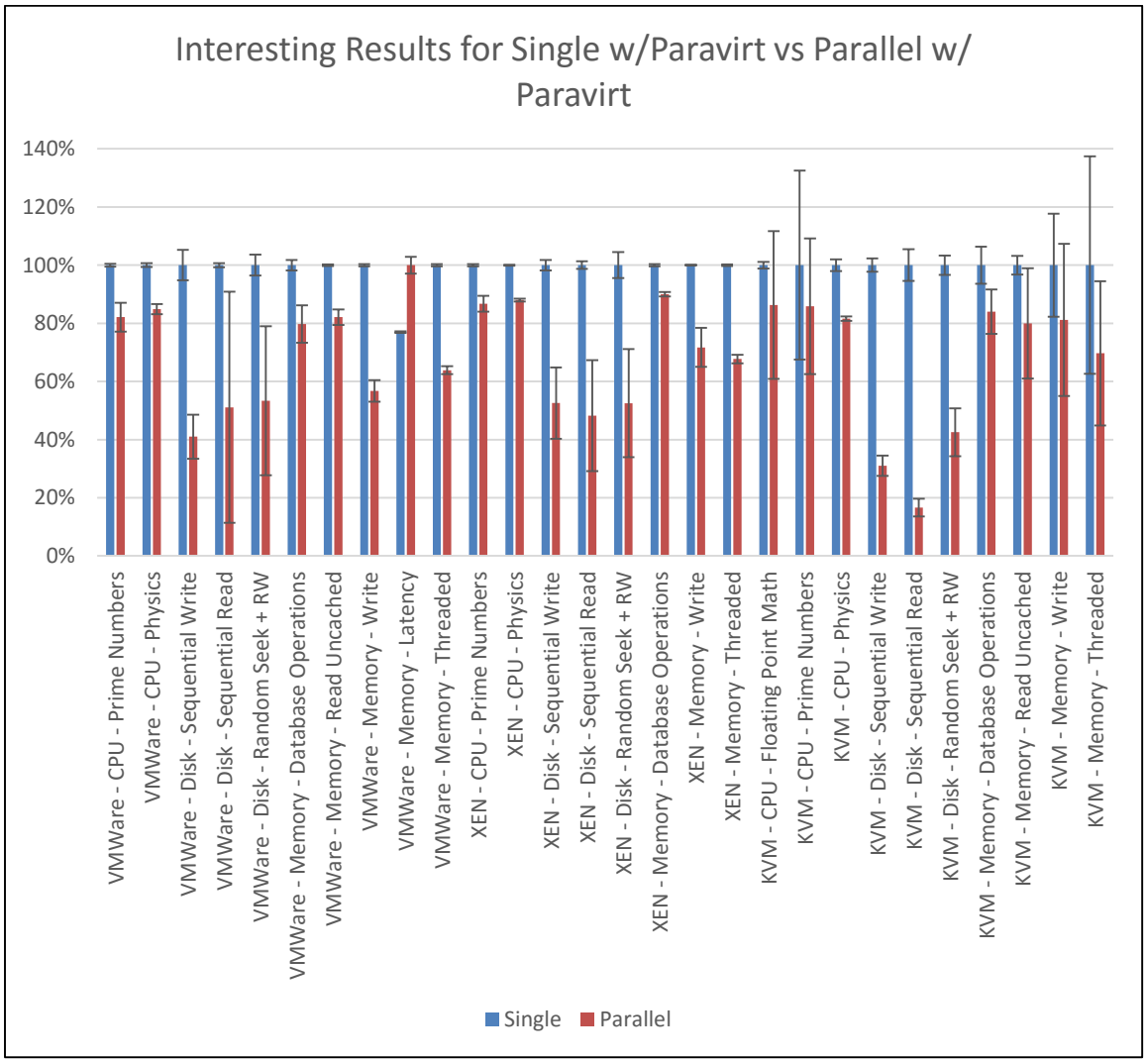


Figure 13: Results for Single with paravirtualization vs. Parallel with paravirtualization

7.2.5 Cross hypervisor comparison for single no modifications (SNM)

For all of the interesting results in this category of tests KVM had the poorest performance. This shows that KVM "out of the box" generally has worse performance than the other two hypervisors (see Figure 14). The two other hypervisors are very close in performance in most of the interesting categories except for the "Random Seek + RW" test in which Xen had a significantly better performance than any other the other hypervisors. Why this is the case is uncertain, perhaps certain optimizations are in place for the default install of Xen that are not in the other hypervisors.

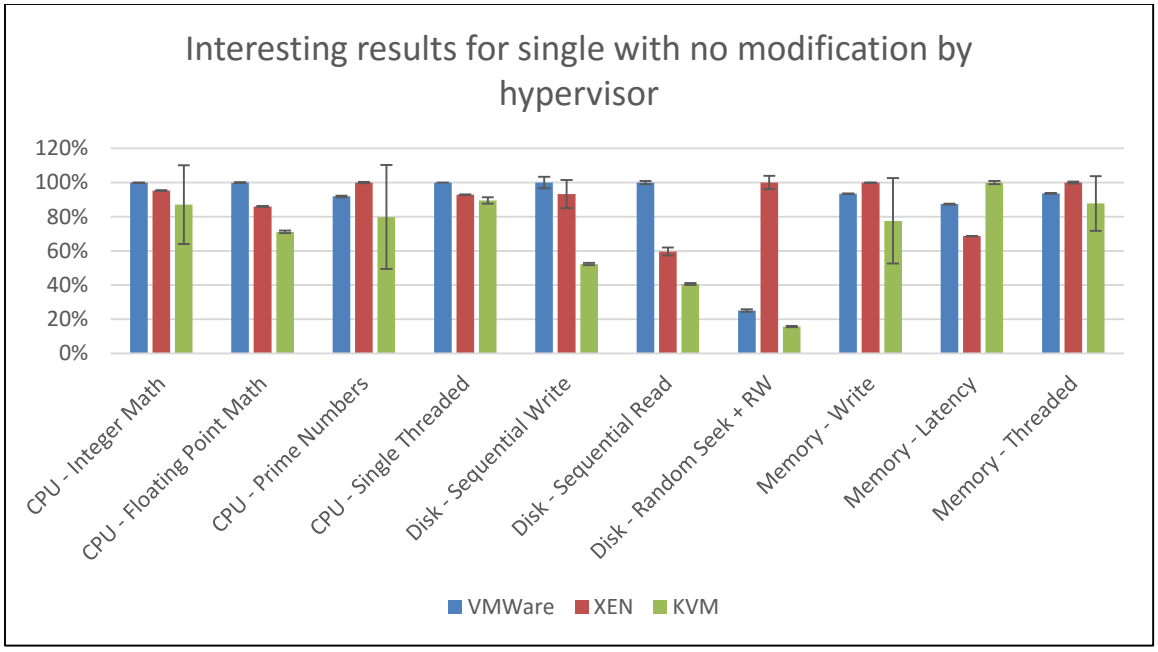


Figure 14: Results for Single no Modifications across Hypervisors

7.2.6 Cross hypervisor comparison for single with paravirtualization (SWP)

This test set has several interesting data points when compared to the previous test configuration. In this test scenario KVM actually performed as well or better than the other hypervisors in most of the interesting results. We see a drastic improvement of disk performance and significant increases in several other tests for KVM in this test scenario. Also several of the CPU tests no longer show significant differences between the hypervisors, indicating that the paravirtualization drivers equalized the performance across the hypervisors for several tests. From these results we can see very minimal differences between hypervisors except for a couple scenarios. (see Figure 15)

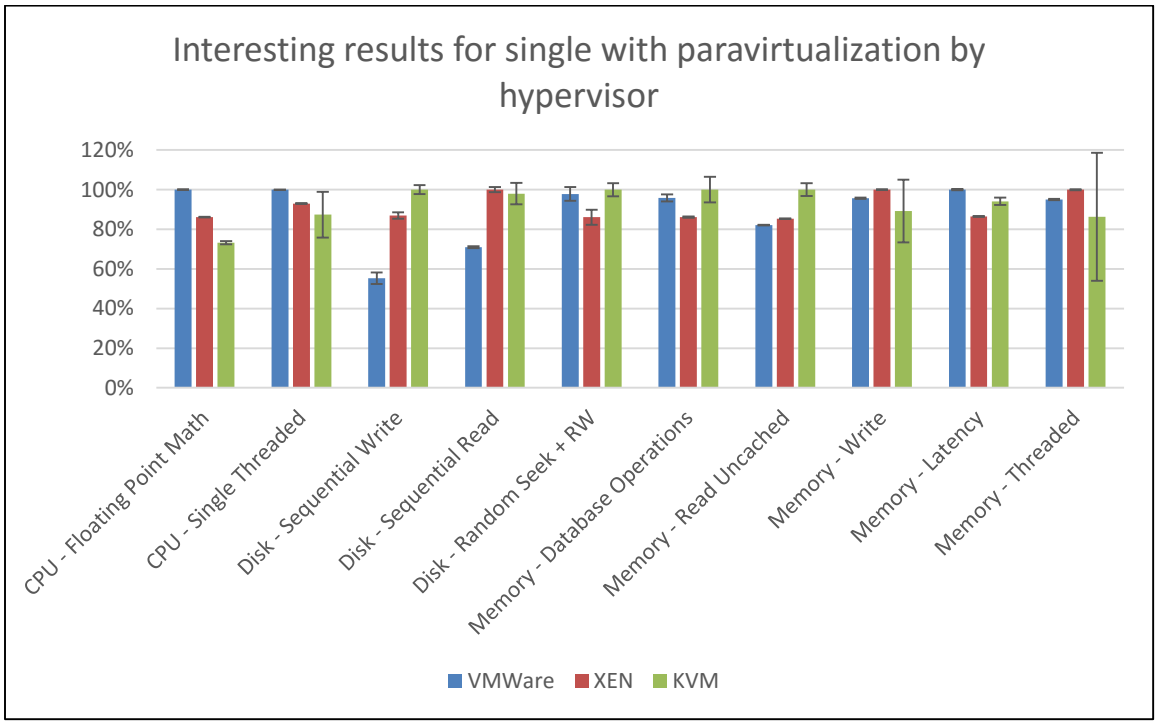


Figure 15: Results for Single with paravirtualization across Hypervisors

7.2.7 Cross hypervisor comparison for parallel no modifications (PNM)

The results for this test scenario are very similar to the single machine no modifications results. KVM shows significantly low disk performance while the other hypervisors stay relatively close. It is interesting to note that the variance in the disk performance tests in both Xen and VMWare are significantly larger than with the single machine results (see Figure 16). This is possibly due to some disk operations on individual guests blocking while others execute. This increase in variance is not seen in the KVM tests which could potentially mean that the algorithm used for disk I/O in KVM has a way of reducing blocking.

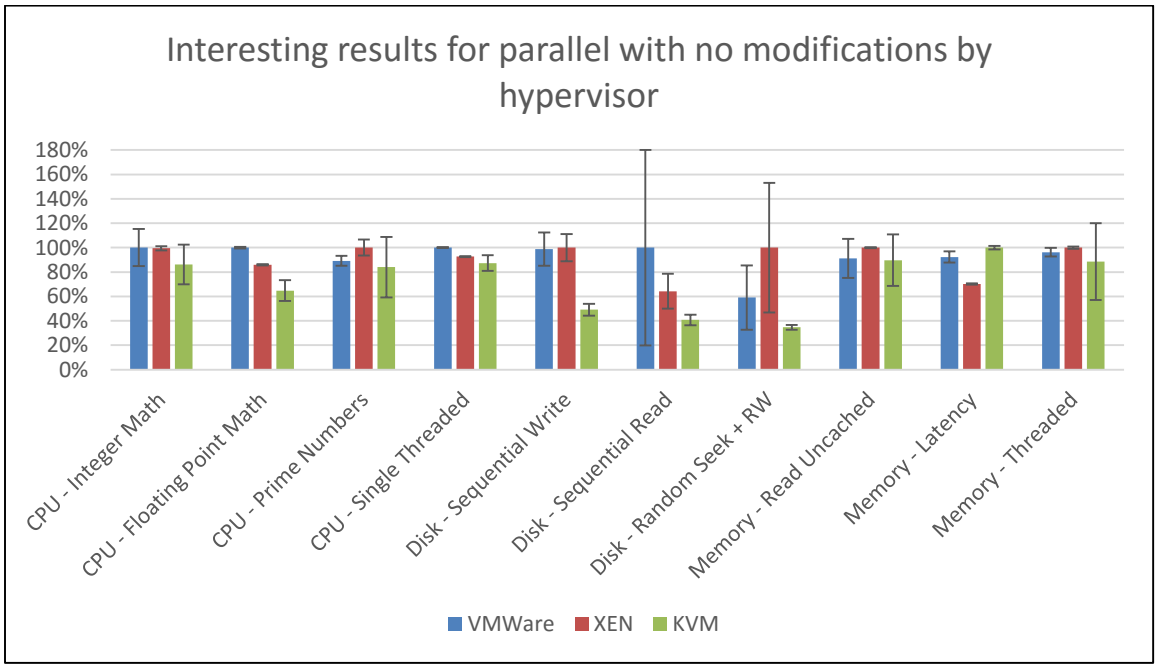


Figure 16: Results for Parallel no Modifications across Hypervisors

7.2.8 Cross hypervisor comparison for parallel with paravirtualization (PWP)

For this comparison similar results to the "SWP by hypervisor" were expected to be seen, in which many of the tests would show a leveling out of performance across all hypervisors. However, in several scenarios, specifically the disk tests, this was not the case. For the sequential read and write tests Xen showed a dramatic increase in performance when compared the other hypervisors. KVM also showed less of an increase in performance than it showed in the SWP configuration indicating that the read performance in a parallel guest machine configuration has not been optimized as much as the other two hypervisors (see Figure 17). It would be interesting to see if this trend continued to appear with additional parallel machine being added.

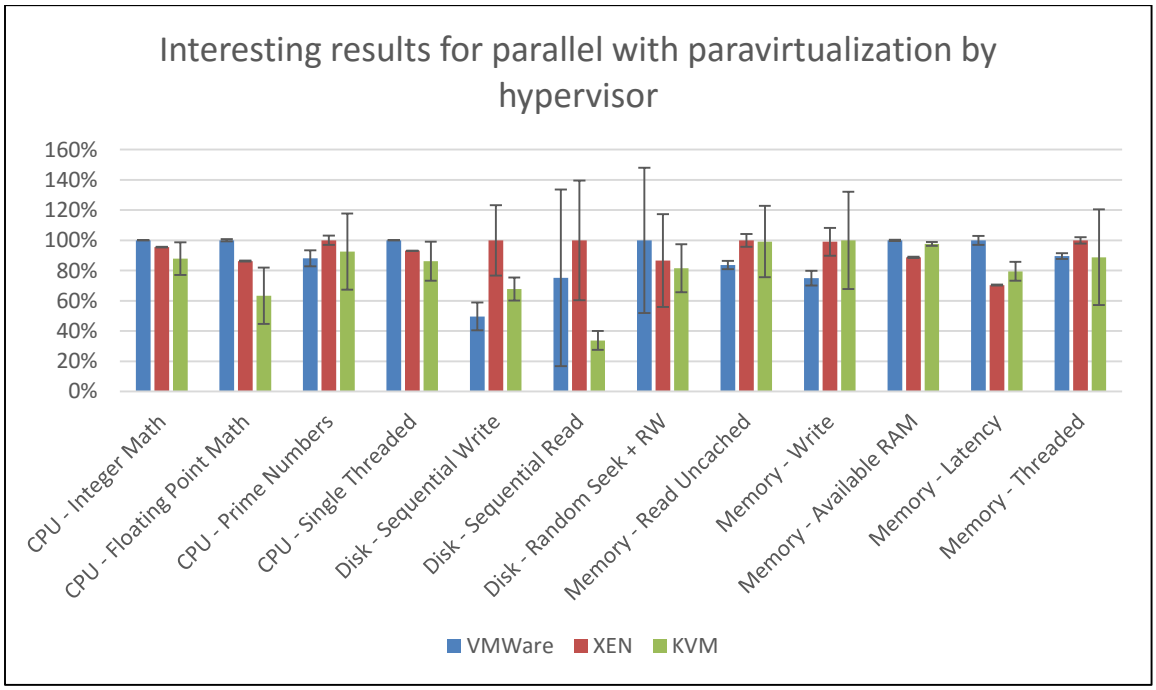


Figure 17: Results for Parallel with paravirtualization across Hypervisors

7.2.9 Network performance

7.2.9.1 Network performance across adapter types

All hypervisors except VMware had interesting results from network performance comparisons. KVM showed the largest improvement from the addition of paravirtualization which performed at 50x to 60x (5000-5900%) when compared to the no modification configuration and 5.1x to 5.9x (510-590%) when compared to the e1000 network adapter configuration. XenServer's paravirtualization configuration performed at 22x (2187%) and 1.1x (112%) when compared to the no modification configuration and the e1000 network adapter configuration respectively for the receiving client, for the sender XenServer performed at 24x (2424%) and 2.3x (233%) respectively. (see Figure 18)

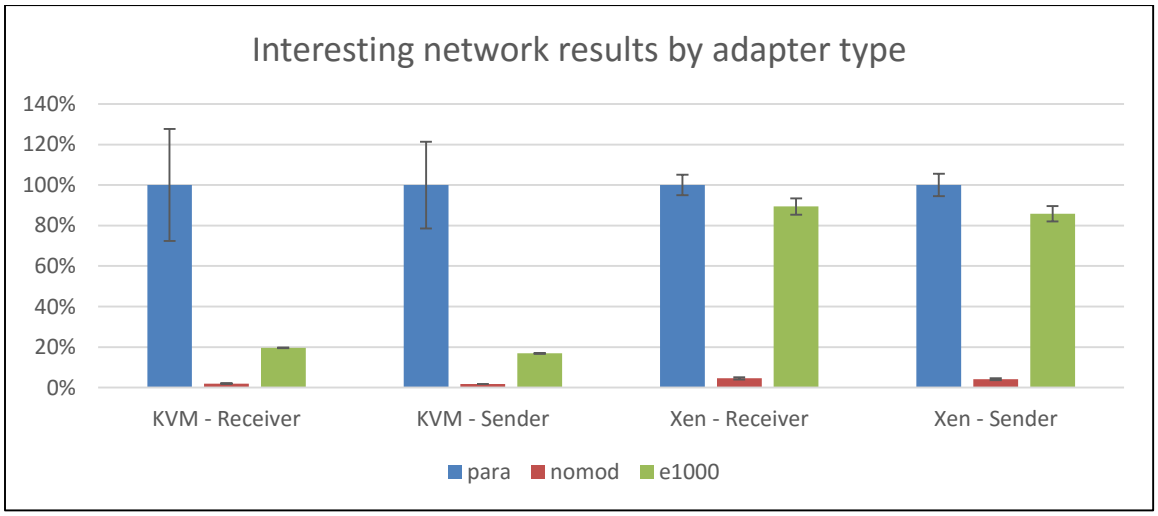


Figure 18: Results for Network testing by adapter type

7.2.9.2 Network performance across hypervisors

VMWare performed the best in every network configuration by at least 1.6x (163%) and up to 100x (9995%). The most interesting result from this comparison is that the VMware tests themselves did not vary much between the types, however, it consistently and significantly out-performed both other hypervisors. The closest results in this category came from the paravirtualization configuration of the receiving client in which VMWare performed at 1.6x (163%) when compare to KVM's performance and 2.7x (272%) when compared to XenServer (see Figure 19). This comparison is very important to the overall perceived performance of the hypervisors and with the ever increasing internet connectivity that the current industry demands will be a deciding factor for a company choosing a hypervisor solution.

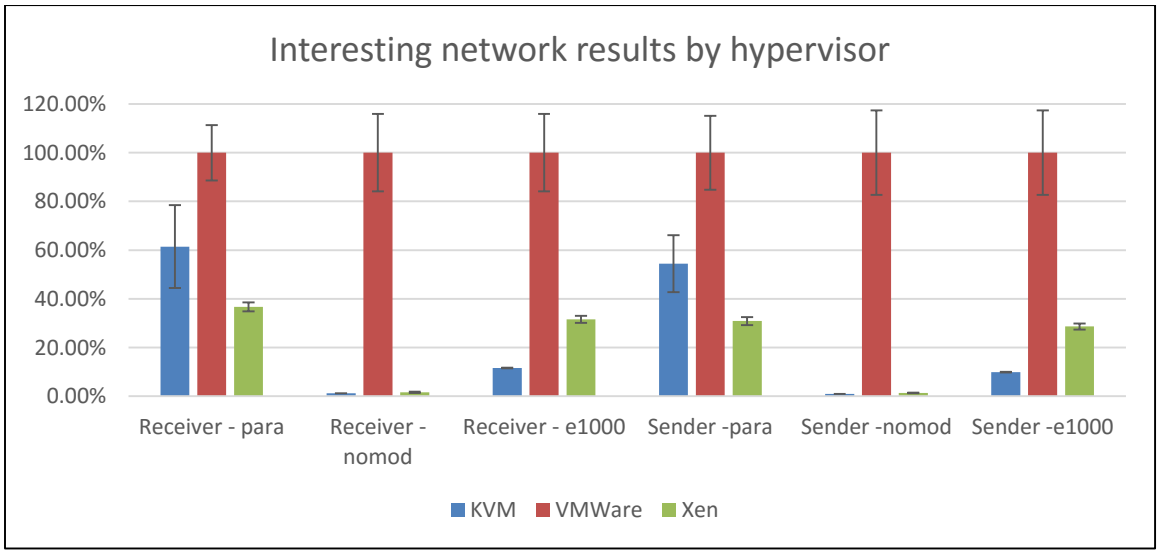


Figure 19: Results for Network tests across hypervisors

Chapter 8

CONCLUSIONS

12,900 data points were collected in this study and 492 comparisons were made, even still there are additional ways of relating this data that could result in additional conclusions being drawn. It is important that the methods of this study be considered in the conclusions. These hypervisors will continue to evolve and in several years the results in this study will not be valid anymore and additional studies will need to be performed. However, the methods presented in this study can be used for any future releases of these or any other hypervisors. With the highly configurable and flexible nature of hypervisors there will be an ever-growing number of components that an end user could add when setting them up. The methods in this paper provide the following key benefits over traditional performance testing:

1) A way of testing and analyzing specific configurations
2) A way of quantifying results based on statistical analysis rather than just means
3) A way of gathering large quantities of data without manual intervention
4) A way of analyzing the performance variation when using n-# of parallel machines executing work simultaneously.
5) A way of comparing results that provides an “inner” view of the performance variation of a configuration. e.g. How adding or removing the configuration affected the hypervisor itself

6) A way of comparing an "outer" view of the performance variation of a configuration. E.g. How adding or removing the configuration affected the hypervisor's relative performance when compared to other hypervisors

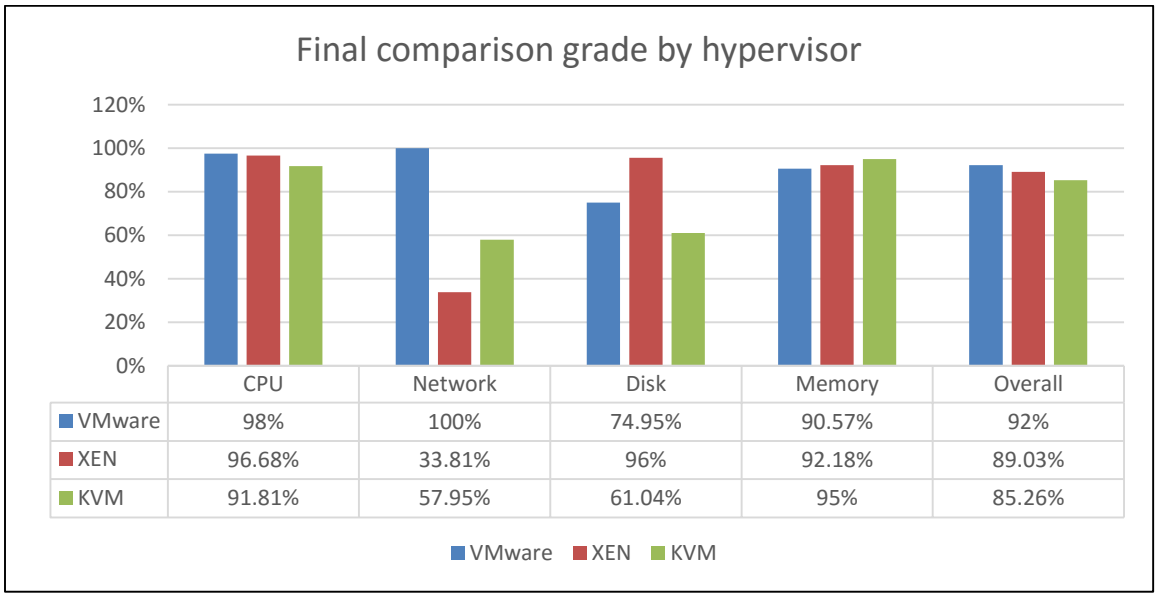
Based on the final results from the performance analysis made, the following conclusions could be implied about the hypervisors tested in this study:

1) The "out of the box" performance of KVM is worse than the "out of the box" performance of the other systems. We can see this in the results from section 7.2.5, KVM's performance in many categories is lower than the other system's. The importance of this conclusion is when looking at many other performance evaluation studies on these hypervisors. Most studies take the default installation of these systems and perform tests without any modifications, which would consistently place the systems that require configurations at the bottom of these tests. In practice, these systems would be configured before use in an actual business setting and would perform closer to the results seen in sections 7.2.7 or 7.2.8. However, the negative aspect of this is that it may steer people away from the technology if the "out of the box" experience is not comparable to the other systems, as that is often the first impression of a technology.
2) The paravirtualization implementation in VMware actually provides very little performance improvement in any category. This can be seen from the results in section 7.2.1 and 7.2.2 in which there are very few categories that show any change in performance when paravirtualization is implemented. This is surprising since usually paravirtualization is considered a way of optimizing performance of

a virtual machine. In VMWare, however, it appears that the method they use for resource consumption is sophisticated enough to not require paravirtualization and still have comparable performance to other hypervisors in the industry. The paravirtualization drivers are likely either drivers optimized for much larger workloads/parallel configurations (such as the balloon driver) or used for administration purposes.

3) XenServer's disk performance showed an unexpected result as well. When adding the paravirtualization drivers the "Random Seek + RW" test showed a significant decrease in performance (see sections 7.2.1 and 7.2.2). However, this decrease brought it closer to the performance of the other two hypervisors. It appears that in the no modification configuration that optimizations were made specifically for workflows similar to the "Random Seek + RW" test. Why this occurs is a subject for another study.
4) KVM, even in the paravirtualization configurations, has an overall decrease in disk performance when using parallel machines (see sections 7.2.3 and 7.2.4). All of the disk tests were roughly 50% or lower when the single configurations were compared to the parallel configurations, meaning that the overall disk performance decreased. This means that for disk I/O heavy workloads KVM would perform better if a single VM was used rather than multiple VMs trying to write to disk simultaneously.
5) The overall results can be seen in (Figure 20) the graph below. VMware had the best overall performance, however, each hypervisor had at least one category in which it had the highest score (even if only by a small percentage). These results

are based on the PWP configuration as it was deemed to be the most likely scenario in a live production environment.



	CPU	Network	Disk	Memory	Overall
VMware	98%	100%	74.95%	90.57%	92%
XEN	96.68%	33.81%	96%	92.18%	89.03%
KVM	91.81%	57.95%	61.04%	95%	85.26%

Figure 20: Overall results across hypervisors

It is important to note that the configurations used in this study are just a small number of what is available for these hypervisors. The ones chosen were based on common industry practices at the time of the study's writing.

Chapter 9

FUTURE WORK

There are near endless possibilities for future work on this subject. Each and every configuration possible could be used in later studies. In particular a couple of configurations that have the potential to be the most interesting are the following:

1. Resource overcommitting – There are several different setups available for resource overcommitment including memory and CPU. Determining if any one hypervisor handles resource sharing worse than the others would be an intensive and interesting study.
2. Higher parallelization count – Testing the performance degradation of running a higher number of machines (four, eight, sixteen, etc.) could provide insight to see how quickly performance degrades as the machine counts increases. However, with the configuration from this study it is likely that the physical system could not handle a higher load than four parallel machines without having major resource contention. A machine with better physical specifications, such as a datacenter server, would need to be used to test the higher machine counts.
3. Solid state disks – Seeing the relative performance of each hypervisor when using SSDs would be an interesting study focus. SSDs are very popular now, even in enterprise companies and datacenters. One or more of the hypervisors might have specific optimizations that increase performance for SSDs.

4. Large Pages – Large Pages as defined in Chapter 4 of this study increases the memory page size for a system. This would be an interesting addition to the current study and could be done with any system configuration. It is important to note that Large Pages/Huge Pages must be enabled for the guest and the host machines to have an effect and cannot be used in parallel with page sharing.
5. Different I/O modes and Disk Caching for KVM – KVM in particular has a large number of configurations that any guest machine can use. Disk caching, for this study was set to "none" to prevent the cache from skewing the final results; however, there are other disk caching modes available such as "writeback" and "writethrough" that both perform differently under different workloads. In addition the I/O mode for a guest can be "native" or "threads" which change how the system writes to the disk. Both of these options could potentially give interesting results.
6. Video/Graphics testing – One subject not touched on in this study is graphics and GPU virtualization. It would be interesting to see the performance differences of the different hypervisor under tests designed to test the GPU directly. PassMark Performance test actually provides these tests and they could be used in a very similar manner as the other tests were in this study.

In addition to different configurations additional hypervisors could also be used, Microsoft's Hyper-V product was excluded from this study merely to limit scope. It and

other hypervisors are very popular in the industry and could be included in further studies.

REFERENCES

Print Publications:

[Barham03]
Barham, Paul, et al. "Xen and the art of virtualization." *Proceedings of the nineteenth ACM symposium on Operating System principles*. New York, NY, USA, 2003. 164-177.

[Che08]
Che, Jianhua, et al. "Performance Measuring and Comparing of Virtual Machine Monitors." *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*. Hangzhou, China, 2008.

[Chierici10]
Chierici, Andrea and Riccardo Veraldi. "A Quantitative Comparison Between Xen and KVM." *17th International Conference on Computing in High Energy and Nuclear Physics*. 2010.

[Cox06]
Cox, Alan L and Willy Zwaenepoel. "Optimizing Network Virtualization in Xen." *Proc. USENIX Annual Technical Conference*. 2006.

[Deshane08]
Deshane, Todd, et al. "Quantitative Comparison of Xen and KVM." *Xen Summit*. Boston, MA, 2008.

[Hwang13]
Hwang, Jinho, Sai Zeng, Frederick Y. Wu, and Tim Wood. "A component-based performance comparison of four hypervisors." *In Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pp. 269-276. IEEE, 2013.

[Hogg10]
Hogg, Robert V and Elliot A. Tannis. "Probability and Statistical Inference." *Pearson Education, Inc* 2010.

[Mohan12]
Mohan, Biju R and Deepak K Damodaran. "Performance Measuring and Comparison of VirtualBox and VMware." *International Proceedings of Computer Science & Information Technology 27*. 2012.

[Motika11]
Motika, Gal and Shlomo Weiss. "Virtio network paravirtualization driver: Implementation and performance of a de-facto standard." *Computer Standards and Interfaces* 34.1 (2011): 36-47.

[Rodriguez12]
Rodriguez-Haro, Fernando, et al. "A summary of virtualization techniques." *Iberoamerican Conference on Electronics Engineering and Computer Science*. 2012.

[Russell08]
Russell, Rusty. *virtio: Towards a De-Facto Standard For Virtual I/O Devices*. Canberra, Australia: SIGOPS Oper. Syst. Rev, 2008.

[Soltesz07]
Soltesz, Stephen, et al. *Container-based Operation System Virtualization: A Scalable, High-performance Alternative to Hypervisors*. Princeton, New Jersey: EuroSys, 2007.

[Suganaya12]
Suganaya, Sridharan. *A Performance Comparison of Hypervisors for Cloud Computing*. Jacksonville, FL: UNF Thesis and Dissertations. Paper 269, 2012. http://digitalcommons.unf.edu/etd/269.

[VMWare06]
VMWare, Inc. *Virtualization Overview*. Palo Alto, CA, 2006. White Paper

[VMWare07A]
VMWare, Inc. *A Performance Comparison of Hypervisors*. Palo Alto, CA, 2007. White Paper

[VMWare07B]
VMWare, Inc. *Understanding Full Virtualization, Paravirtualization, and Hardware Assist*. Palo Alto, CA, 2007. White Paper

[VMWare09A]
VMware, Inc. *Performance Evaluation of VMXNET3 Virtual Network Device*. Palo Alto, CA: VMware, Inc, 2009. White Paper

[VMWare09B]
VMware, Inc. *What is new in VMWare vShpere 4: Storage*. Palo Alto, CA: VMware, 2009. White Paper

[VMWare11]
VMWare, Inc. *Performance Best Practices for VMware vSphere 5.0*. Palo Alto, CA, 2007-2011. White Paper

[Xianghua08]
Xianghua, Xu, et al. "Quantifying Performance Properties of Virtual Machine." *International Symposium on Information Science and Engineering*. 2008.
Electronic Sources:

[IBM12A]
IBM Corporation. *Best practices for KVM*. 2012. https://www-01.ibm.com/support/knowledgecenter/linuxonibm/liaat/liaatbestpractices_pdf.pdf, last accessed September 26. 2015

[IBM12B]
IBM. *On the Origin of KVM*. 13 Fedruary 2012. IBM Software Defined. https://www.ibm.com/developerworks/community/blogs/ibmvirtualization/entry/on_the_origin_of_kvm13?lang=en, last accessed September 26, 2015

[Kumar10]
M. Kumar, S. Roberts and C. Kawalek, "The Most Complete and Integrated," Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA, 2010. http://www.oracle.com/us/technologies/virtualization/virtualization-strategy-wp-183617.pdf, last accessed September 26, 2015

[Microsoft08]
Microsoft, Inc. *How to Use NTttcp to Test Network Performance*. 15 April 2008. http://msdn.microsoft.com/en-us/windows/hardware/gg463264.aspx, last accessed September 26, 2015

[RedHat14]
Red Hat, Inc. *5.4. Huge Pages and Transparent Huge Pages (THP)*. 2014. Red Hat, Inc. 5th. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Tuning_and_Optimization_Guide/sect-Virtualization_Tuning_Optimization_Guide-Memory-Huge_Pages.html, last accessed September 26, 2015

[Linux07]
Linux 2.6.20 Release Notes. 5 February 2007. http://kernelnewbies.org/Linux_2_6_20, last accessed September 26, 2015

[Xen15]
Paravirtualized SCSI: XEN PVSCSI. n.d. http://wiki.xen.org/wiki/Paravirtualized_SCSI, last accessed September 26, 2015

APPENDIX A

XENSERVER HOST CONFIGURATIONS

After initial tests were conducted on the no modification configuration of XenServer the following was done to the host system to allow the guest machines to use the e1000 adapter type:

1) Move the original qemu file XenServer uses to boot virtual machines:

   mv /usr/lib/xen/bin/qemu-dm /usr/lib/xen/bin/qemu-dm.orig

2) Create new file '/usr/lib/xen/bin/qemu-dm' with the following contents:

   ```
   #!/bin/bash
   oldstring=$@
   newstring=${oldstring/=rtl8139/=e1000}
   exec /usr/lib/xen/bin/qemu-dm.orig $newstring
   ```

3) Enable execute for new file: chmod 755 /usr/lib/xen/bin/qemu-dm

This process overrides the default network adapter that XenServer uses when booting a virtual machine. With this in place any virtual machines that would normally be booted with the rtl8139 network adapter are now booted with the e1000 network adapter.

APPENDIX B

KVM HOST CONFIGURATIONS

The base install for the system used to host KVM is a little more complicated since Fedora was used instead of a dedicated system like VMWare and XenServer provide. With that in mind a bit of modification to the host was required to reduce the overhead of running a full version of Linux in parallel with KVM. The changes did not include any direct modifications to the guest machines. They are used to get the Fedora install match the other hypervisors a little more closely. The following is how the host machine was setup:

1) The '/home' partition was removed from the default partition schema. This is done because we do not have a need for a '/home' partition since the machine will be dedicated to hosting guest machines.
2) No additional users were added to the system, the 'root' user is all that was required for administration.
    a. To allow 'root' to log into the GUI the following needs to be removed or commented out in the '/etc/pam.d/lxdm' file: auth required pam_succeed_if.so user != root quiet
3) KVM and the management tools associated to KVM (e.g. virt-manager) were installed along with a Network Time Protocol (NTP) service to allow the clocks on the Host and Guest machines to sync to the same NTP servers as VMWare and XenServer.

a. Install applications: yum -y install kvm virt-manager libvirt ntp
b. Configure NTP: Change server IPs in /etc/ntp.conf to be trusted NTP server(s)
c. Start the NTP service: systemctl start ntpd.service
d. Enable the NTP service so that it starts with the Host: systemctl enable ntpd.service

4) Applications that were not required for KVM to function were removed
    a. yum remove clipit audit smartmontools gxine pidgin osmo abiword gnumeric leafpad xpad gnomebaker asunder sylpheed transmission
5) System was changed to only boot to run level 3, which does not start the GUI and other ancillary applications.
    a. unlink /etc/systemd/system/default.target
    b. ln -s /lib/systemd/system/runlevel3.target default.target
6) By default both VMWare and XenServer utilize Bridged networking (described above in the paper) for their guest machines on standard installation, however, KVM defaults to NAT. The normal service for managing network interfaces on Fedora is called NetworkManager, but this service does not properly provide the support for Bridged networking that is required for this configuration. So the NetworkManager service was disabled and a networking service that supports network bridges was enabled.
    a. systemctl stop NetworkManager.service
    b. disable NetworkManager.service
    c. systemctl start network.service

d. systemctl enable network.service

7) VMWare and XenServer both also provide graphical remote management tools (e.g VMWare v-Sphere Client and Citrix XenCenter), KVM does not provide this out of the box, so a slight modification is required for ease of administration. Linux has very good support for Virtual Network Computing (VNC) which is a way to access the system's user interface remotely. Utilizing this it provides us with an easy and familiar way to remotely manage the Host and Guests.

   a. Install the VNC server:

      i. yum -y install tigervnc-server

   b. Set the VNC service to listen on port 5900:

      i. cp /lib/systemd/system/vncserver@.service

      ii. /etc/systemd/system/vncserver@:0.service

   c. replace <USER> tags in vncserver@:0.service just created with 'root' user. This sets the default login for VNC as the 'root' user.

   d. Add an exception to the Host's firewall to allow VNC connections publicly

      i. firewall-cmd --permanent --zone=public --add-service vnc-server

   e. Configure the VNC password by running: vncpasswd

   f. Reload the service daemon: systemctl daemon-reload

   g. Enable the VNC service: systemctl enable vncserver@:0.service

   h. Restart the system so that the changes take affect

   i. Run the 'vncserver' command to verify it has been activated correctly

APPENDIX C

GUEST CONFIGURATIONS

The operating system used for this research study was Microsoft Windows Server 2008 R2 SP1.

Each guest was configured with the following on each hypervisors:

1) 2 vCPUs each pinned to their own logical core.
2) 1536 MB of Memory
3) 40GB of disk space persistent and allocated up front

For KVM two additional steps were performed:

4) A bridged network interface was added through the Virtual Machine Manager (virt-manager) and used as the source device for each guest machine's virtual network interface.
5) Disk cache mode was set to 'none'

Once started each machine had the following steps applied:

1) Administrator password was set
2) Windows activation key was set
3) IE ESC was disabled
4) PassMark Performance 8.0 was downloaded and installed.
5) PassMark Performance 8.0 was opened and license key was installed
6) Net TCP Testing Tool was downloaded and installed
    a. Standard install using "Everyone" configuration

b. NTttcp_x64.exe was renamed to NTttcpr.exe and NTttcps.exe

c. InBound Firewall rule added to allow connections for NTttcpr.exe

d. OutBound Firewall rule added to allow connections for NTttcps.exe

7) Microsoft .Net 4.0 was downloaded and installed (required for ParallelExecution v1.0)

8) ParallelExecution v1.0 was downloaded and extracted to 'C:\Users\Administrator\Desktop\ParallelExecution\'

9) Folder named 'results' was added to 'C:\'

10) Sharing privileges added to allow results to be mapped as a network drive (Properties -> Sharing -> Advanced Sharing -> Share this Folder)

APPENDIX D

PHYSICAL SYSTEM SPECIFICATIONS

1) Hardware Specifications
    a. Model: Dell OptiPlex 980
    b. CPU Cores: 4CPUs x 2.795 GHz
    c. Processor: Intel Core i7 860, Processor Sockets: 1, Processor Cores per Socket: 4, Logical Processors: 8, Hyperthreading enabled
    d. Memory: 4019.36 MB DDR3
    e. Disk Storage: 460.75 GB, 7200 RPM, 16MB Cache, Format: VMFS 5.54 - 1MB Block Size
    f. Network: Intel Corporation 82578DM Gigabit Network Connection
2) Software Versions
    a. Hypervisors
        i. VMware ESXi - 5.1.0, Release build-838463
        ii. XenServer - 6.2.0-70446ac
        iii. KVM - QEMU version 1.4.2 on Fedora 19 (3.9.5-301.fc19.x86_64)
    b. Testing Tools
        i. CPU, RAM, Disk: PassMark Performance 8.0
        ii. Network: NT Testing TCP Tool v3.0

APPENDIX E

PERFORMANCE METRICS

1) PassMark Performance Test 8.0 (Taken from 'Help' files for the application):
    a. CPU Mark – Measures CPU performance.
        i. Integer Math – In Millions of Operations per second – Tests how quickly the CPU can perform mathematical integer operations. An integer is considered a whole number with no fractional part. The test uses integer buffers totaling about 240kb per core.
        ii. Floating Point Math – In Millions of operations per second – Same operations as integer math but using floating point numbers. Uses the same size memory buffers as integer math per core.
        iii. Prime Number Test – In Millions of primes per second – Tests how quickly the CPU can search for prime numbers. Uses the Sieve of Atkin formula with a limit of 32 million.
        iv. Multimedia Instructions – In Millions of Matrices per second – Measures the Streaming SIMD Extensions (SSE) capabilities of a CPU. SSE is a set of CPU instructions that have been introduced into CPUs to enable blocks of data to be processed at higher speeds. Test measures the number of times a 4x4 matrix can be multiplied by a 4 dimensional vector per second, with the vector represented by a 128-bit floating point number and the matrix being represented by 4 128-bit floating point numbers.

v. Compression Test – In Kbytes processed per second – Tests speed in which CPU can compress blocks of data.

vi. Encryption Test – In Mbytes transferred per second – Tests speed in which the CPU can encrypt data using several encryption techniques, Including TwoFish, AES, Salsa20, and SHA256.

vii. Physics Test – In Frames per second – Uses basic Physics engine to test how quickly a CPU can calculate physics interactions of several hundred object colliding.

viii. Sorting Test – In Thousands of String per second – uses qqSort algorithm to test how quickly the CPU can sort strings.

ix. Single Core Test – In Millions of operations per second – Uses floating point, string sorting, and data compression tests to test the performance of a single core.

b. Memory Mark – Measures memory performance.

i. Database Operations – In thousands of operations per second - This makes heavy use of C++ STL containers to test the performance of memory in maintaining large structures of data like that in a database.

ii. Read Cached – In Mbytes transferred per second - This test measures the time taken to read a small block of memory. The block is small enough to be held entirely in cache (if one is present)

iii. Read Uncached – In Mbytes transferred per second - This test measures the time taken to read a large block of memory when the block is too large to be held in cache.

iv. Write – In Mbytes transferred per second - This test measures the time taken to write information into memory.

v. Latency – In Nanoseconds - Time it takes for a single byte of memory to be transferred to the CPU for processing. Measured in nanoseconds, lower values are better.

vi. Threaded Test – In Mbytes transferred per second - Nearly identical operations to the Read Uncached test, however being performed by two separate processes simultaneously to test how well memory copes with multiple concurrent accesses.

c. Disk Mark – Measures disk drive performance.

i. Sequential Read – In Mbytes transferred per second - A large test file is created on the disk under test. The file is read sequentially from start to end.

ii. Sequential Write – In Mbytes transferred per second - A large file is written to the disk under test. The file is written sequentially from start to end.

iii. Random Seek + RW – In Mbytes transferred per second - A large test file is created on the disk under test. The file is read randomly; a seek is performed to move the file pointer to a random position in the file, a 16KB block is read or written then another seek is performed. The amount of data actually transferred is highly dependent on the disk seek time.

d. 2D & 3D Graphics – PassMark Performance Test has several tests for 2D and 3D graphics but these will not be used in the paper.

2) NT TCP Testing Tool – “NTttcp is a multithreaded, asynchronous application that sends and receives data between two or more endpoints and reports the network performance for the duration of the transfer. It is essentially a Winsock-based port of the ttcp tool that measures networking performance in terms of bytes transferred per second and CPU cycles per byte.” [Microsoft08]

APPENDIX F

LINQPAD SCRIPT

```
void Main()
{
  //NETWORK TESTS
  var fileIdentifiers = new List<string>(){
"results_remote_bridge_para", "results_remote_bridge_nomod",
"results_remote_bridge_e1000"  };

  var networkTests = new List<string>();

  //Add mappings for senders and receivers
  networkTests.AddRange(fileIdentifiers.Select(f => "Receiver - " +
f));
  networkTests.AddRange(fileIdentifiers.Select(f => "Sender - " + f));

  var rootNetDirectory = "C:\\Users\\Sean
McAdams\\Dropbox\\Thesis\\Network Tests";

  var files = NetworkResultsParser.GetTxtFilesInDirectoryTree(new
DirectoryInfo(rootNetDirectory));

  var networkResults = new OneDimensionalDataSet<double?>();

  //For each file get throughput and add to results.
    foreach(var file in files){
    string columnForFile = null;
    foreach(var fileIdentifier in fileIdentifiers){
      if(file.Name.Contains(fileIdentifier)){
        columnForFile = file.Directory.Name + " - " + fileIdentifier;
        break;
      }
    }
    if(columnForFile == null){
      ("INVALID FILE - " + file.Name).Dump();
    } else {
      networkResults.AddItem(columnForFile,
NetworkResultsParser.GetThroughput(file.FullName));
    }
  }

  //Copy results from nomod into e1000 column since VMWare defaults to
e1000
  networkResults.GetItems("VMWare - Receiver -
results_remote_bridge_nomod").ForEach(r =>
networkResults.AddItem("VMWare - Receiver -
results_remote_bridge_e1000", r));
```

```
  networkResults.GetItems("VMWare - Sender -
results_remote_bridge_nomod").ForEach(r =>
networkResults.AddItem("VMWare - Sender - results_remote_bridge_e1000",
r));


  //DISK, MEMORY, AND CPU TESTS
  var rootPath = "C:\\Users\\Sean McAdams\\Dropbox\\Thesis\\Parallel
Perf Tests\\";
  var rootPath2 = "C:\\Users\\Sean McAdams\\Dropbox\\Thesis\\Perf
Tests\\";

  var folderDictionary = new Dictionary<string, string[]>(){
    {"VMWare", new string[]{"Results-VMWare-1", "Results-VMWare-2"}},
    {"XEN", new string[]{"Results-XEN-1", "Results-XEN-2"}},
    {"KVM", new string[]{"Results-KVM-1", "Results-KVM-2"}},
  };

  var fileDictionary = new Dictionary<string, string>(){
    {"{0}results-{1}CPU_INTEGERMATH.csv", "CPU - Integer Math"},
    {"{0}results-{1}CPU_FLOATINGPOINTMATH.csv", "CPU - Floating Point
Math"},
    {"{0}results-{1}CPU_PRIME.csv", "CPU - Prime Numbers"},
    {"{0}results-{1}CPU_SSE.csv", "CPU - Extended Instructions (SSE)"},
    {"{0}results-{1}CPU_COMPRESSION.csv", "CPU - Compression"},
    {"{0}results-{1}CPU_ENCRYPTION.csv", "CPU - Encryption"},
    {"{0}results-{1}CPU_PHYSICS.csv", "CPU - Physics"},
    {"{0}results-{1}CPU_SORTING.csv", "CPU - Sorting"},
    {"{0}results-{1}CPU_SINGLETHREAD.csv", "CPU - Single Threaded"},
    {"{0}results-{1}DI_WRITE.csv", "Disk - Sequential Write"},
    {"{0}results-{1}DI_READ.csv", "Disk - Sequential Read"},
    {"{0}results-{1}DI_RANDOM.csv", "Disk - Random Seek + RW"},
    {"{0}results-{1}ME_ALLOC_S.csv", "Memory - Database Operations"},
    {"{0}results-{1}ME_READ_CACHED.csv", "Memory - Read Cached"},
    {"{0}results-{1}ME_READ_UNCACHED.csv", "Memory - Read Uncached"},
    {"{0}results-{1}ME_WRITE.csv", "Memory - Write"},
    {"{0}results-{1}ME_LARGE.csv", "Memory - Available RAM"},
    {"{0}results-{1}ME_LATENCY.csv", "Memory - Latency"},
    {"{0}results-{1}ME_THREADED.csv", "Memory - Threaded"}
  };

  //File Name Format: results-TEST_NAME.csv (forgot to append 'multi-'
during testing)
  var parallelFileMappings = new List<FileMapping>();
  foreach(var key in folderDictionary.Keys){
    foreach(var fileKeyItem in fileDictionary.Keys){
      var fileKey = string.Format(fileKeyItem, "", "");
      //One folder for each machine
      parallelFileMappings.Add(new FileMapping(key, rootPath +
folderDictionary[key][0] + "\\" + fileKey, fileDictionary[fileKeyItem],
key + " - " + fileDictionary[fileKeyItem]));
      parallelFileMappings.Add(new FileMapping(key, rootPath +
folderDictionary[key][1] + "\\" + fileKey, fileDictionary[fileKeyItem],
key + " - " + fileDictionary[fileKeyItem]));
    }
  }
```

```
  //File Name Format: nomod-results-multi-TEST_NAME.csv
  var parallelNoModsFileMappings = new List<FileMapping>();
  foreach(var key in folderDictionary.Keys){
    foreach(var fileKeyItem in fileDictionary.Keys){
      var fileKey = string.Format(fileKeyItem, "nomod-", "multi-");
      //One folder for each machine
      parallelNoModsFileMappings.Add(new FileMapping(key, rootPath +
"NoMod-" + folderDictionary[key][0] + "\\" + fileKey,
fileDictionary[fileKeyItem], key + " - " +
fileDictionary[fileKeyItem]));
      parallelNoModsFileMappings.Add(new FileMapping(key, rootPath +
"NoMod-" +  folderDictionary[key][1] + "\\" + fileKey,
fileDictionary[fileKeyItem], key + " - " +
fileDictionary[fileKeyItem]));
    }
  }

  //File Name Format: results-single-TEST_NAME.csv
  var singleFileMappings = new List<FileMapping>();
  foreach(var key in folderDictionary.Keys){
    foreach(var fileKeyItem in fileDictionary.Keys){
      var fileKey = string.Format(fileKeyItem, "", "single-");
      singleFileMappings.Add(new FileMapping(key, rootPath2 +
folderDictionary[key][0] + "\\" + fileKey, fileDictionary[fileKeyItem],
key + " - " + fileDictionary[fileKeyItem]));
    }
  }

  //File Name Format: nomods-results-single-TEST_NAME.csv
  var singleNoModsFileMappings = new List<FileMapping>();
  foreach(var key in folderDictionary.Keys){
    foreach(var fileKeyItem in fileDictionary.Keys){
      var fileKey = string.Format(fileKeyItem, "nomods-", "single-");
      singleNoModsFileMappings.Add(new FileMapping(key, rootPath2 +
"NoMods-" + folderDictionary[key][0] + "\\" + fileKey,
fileDictionary[fileKeyItem], key + " - " +
fileDictionary[fileKeyItem]));
    }
  }

  //method used to convert the string values of each cell of the csv
into the doubles that i want
  Func<string, double?> converter = (value) => {
    double result;
    return double.TryParse(value, out result) ? result : (double?)
null;
  };

  //Get csvs into datastructures for easy viewing and processing
  var parallelNoModsResults =
PerformanceResultsParser.ParseFiles("Parallel No Mods",
parallelNoModsFileMappings, converter);
  var parallelResults = PerformanceResultsParser.ParseFiles("Parallel",
parallelFileMappings, converter);
  var singleNoModsResults = PerformanceResultsParser.ParseFiles("Single
No Mods" , singleNoModsFileMappings, converter);
```

```
  var singleResults = PerformanceResultsParser.ParseFiles("Single" ,
singleFileMappings, converter);

  //View the resulting data
  (new List<Tuple<string, DataTable>> () {
    Tuple.Create("Parallel No Mods Results",
parallelNoModsResults.AsDataTable()),
    Tuple.Create("Parallel Results w/ Paravirt",
parallelResults.AsDataTable()),
    Tuple.Create("Single No Mods Results",
singleNoModsResults.AsDataTable()),
    Tuple.Create("Single Results w/ Paravirt",
singleResults.AsDataTable()),
    Tuple.Create("Network Results", networkResults.AsDataTable())
  }).Dump("Result Tables");

  DataCompare.DoCrossCompare(new List<OneDimensionalDataSet<double?>>()
{ parallelNoModsResults, parallelResults }).Dump("Parallel No Mods vs
Parallel w/ Paravirt", 10);

  DataCompare.DoCrossCompare(new List<OneDimensionalDataSet<double?>>()
{ singleNoModsResults, singleResults }).Dump("Single No Mods vs Single
w/ Paravirt", 10);

  DataCompare.DoCrossCompare(new List<OneDimensionalDataSet<double?>>()
{ singleNoModsResults, parallelNoModsResults }).Dump("Single No Mods vs
Parallel No Mods", 10);

  DataCompare.DoCrossCompare(new List<OneDimensionalDataSet<double?>>()
{ singleResults, parallelResults }).Dump("Single w/ Paravirt vs
Parallel w/ Paravirt", 10);

  DataCompare.DoInnerCompare(singleNoModsResults,
fileDictionary.Values).Dump("Single No Mods cross Hypervisors", 10);

  DataCompare.DoInnerCompare(singleResults,
fileDictionary.Values).Dump("Single w/ Paravirt cross Hypervisors",
10);

  DataCompare.DoInnerCompare(parallelNoModsResults,
fileDictionary.Values).Dump("Parallel No Mods cross Hypervisors", 10);

  DataCompare.DoInnerCompare(parallelResults,
fileDictionary.Values).Dump("Parallel w/ Paravirt cross Hypervisors",
10);

  //NETWORK RESULTS - Output stored differently from above so different
manipulation needs to be done.

  //split network results by type so that we can compare all 3
configurations by hypervisor
  var networkResultsSplit = fileIdentifiers.Select(i => (new
OneDimensionalDataSet<double?>() {
                                SetName =
i.Substring(i.LastIndexOf("_") + 1),
```

```
                                    Data = networkResults.Data.Where(d =>
d.Key.Contains(i)).ToDictionary(k => k.Key.Substring(0,
k.Key.IndexOf("- " + i)), v => v.Value)
                                    })).ToList();

  DataCompare.DoCrossCompare(networkResultsSplit).Dump("Network No Mods
and Network w/ Paravirt", 10);

  DataCompare.DoInnerCompare(networkResults,
networkTests).Dump("Network No Mods and Network w/ Paravirt cross
Hypervisors", 10);

}

public class ComparisonResult
{
  public bool SetsAreDifferent { get; set; }
  public TwoDimensionalDataSet<double> SetComparison { get; set; }
  public TwoDimensionalDataSet<object> TTestResults { get; set; }
}
public static class DataCompare
{
  static DataCompare(){
    RDotNet.REngine.SetEnvironmentVariables();
  }

  //column names are in the format: "%%setname%% - %%Test%%"
  public static string GetName(string value, string test){
    return value.Substring(0, value.IndexOf(test)).Replace(" ",
"").Replace("-", "");
  }

  //Compares multiple data sets accross each other, matching columns
are found and compared
  public static List<Tuple<string, DataTable>>
DoCrossCompare(List<OneDimensionalDataSet<double?>> resultsSets){
    if(resultsSets.Select(x => x.SetName).Distinct().Count() <
resultsSets.Count){
      throw new ArgumentException("All result sets must have a unique
set name.");
    }

    var stdDev = new TwoDimensionalDataSet<double>();
    var comparisonResults = new OneDimensionalDataSet<DataTable>();
    var comparisonGrades = new TwoDimensionalDataSet<object>();
    var interestingResults = new TwoDimensionalDataSet<object>();
    var interestingResultsStdev = new TwoDimensionalDataSet<object>();

    var testList = new List<string>();
    foreach (var resultSet in resultsSets)
    {
      testList.AddRange(resultSet.Data.Keys);
    }

    foreach(var test in testList.Distinct()){
      List<List<int>> winners = new List<List<int>>();
      var allMeans = new Dictionary<string, double>();
```

```
      //compare each item with each other so if A, B and C then we test
A vs B, A vs C, and B vs C
      //If we have A, B, C, and D then we test A vs B, A vs C, A vs D,
B vs C, B vs D, C vs D
      for(int i = 0; i < resultsSets.Count; i++){
        for(int j = i + 1; j < resultsSets.Count; j++){
          var comparisonResult =
DataCompare.CompareDataSets(string.Format("{0} - {1} vs {2}", test,
resultsSets[i].SetName, resultsSets[j].SetName),
                            resultsSets[i].GetItems(test).Where(x =>
x.HasValue).Select(x => x.Value).ToList(),
                            resultsSets[i].SetName,
                            resultsSets[j].GetItems(test).Where(x =>
x.HasValue).Select(x => x.Value).ToList(),
                            resultsSets[j].SetName);

          stdDev.SetItem(test, resultsSets[i].SetName,
comparisonResult.SetComparison.GetItem(resultsSets[i].SetName,
"Standard Deviation"));
          stdDev.SetItem(test, resultsSets[j].SetName,
comparisonResult.SetComparison.GetItem(resultsSets[j].SetName,
"Standard Deviation"));
          comparisonResults.AddItem(test,
comparisonResult.SetComparison.AsDataTable());
          comparisonResults.AddItem(test,
comparisonResult.TTestResults.AsDataTable());

          allMeans[resultsSets[i].SetName] =
comparisonResult.SetComparison.GetItem(resultsSets[i].SetName, "Mean");
          allMeans[resultsSets[j].SetName] =
comparisonResult.SetComparison.GetItem(resultsSets[j].SetName, "Mean");

          //Set this round's winner(s)
          if(comparisonResult.SetsAreDifferent) {//Not a Tie
            winners.Add(new List<int>() {
allMeans[resultsSets[i].SetName] > allMeans[resultsSets[j].SetName] ? i
: j });
          } else { //Tie
            winners.Add(new List<int>() { i, j });
          }

        }
      }

      //In order to truly have a winner they must have won or tied
every round with the other items
      var winsOrTiesRequired = resultsSets.Count - 1;
      var trueWinners = winners.SelectMany(w => w).GroupBy(w =>
w).Where(g => g.Count() >= winsOrTiesRequired).Select(g =>
g.Key).ToList();

      var maxMean = new List<double>(allMeans.Values).Max();

      //Set is interesting if the grade is < 90% of the max and someone
lost
```

```
      var isInteresting = allMeans.Values.Any(v => v / maxMean < .9) &&
resultsSets.Count != trueWinners.Count;

      //Build grade based on largest mean, grade = ((mean / maxMean) *
100)%
      //If set is interesting, add it to the interesting results
      foreach(var resultSet in resultsSets){
        var setGrade = Math.Round((allMeans[resultSet.SetName] /
maxMean) * 100.0, 2) + "%";
        comparisonGrades.SetItem(resultSet.SetName, test, setGrade);
        if(isInteresting){
          var stdDevGrade = Math.Round((stdDev.GetItem(test,
resultSet.SetName) / maxMean) * 100.0, 2) + "%";
          //interestingResults.SetItem(resultSet.SetName, test,
(Math.Truncate(allMeans[resultSet.SetName] * 1000.0) / 1000.0) + " (" +
setGrade + ")");
          interestingResults.SetItem(resultSet.SetName, test,
setGrade);
          interestingResultsStdev.SetItem(resultSet.SetName, test,
stdDevGrade);
        }
      }

      //Set the winner row for this test based on the found winners
      comparisonGrades.SetItem("Winner", test, string.Join(" And ",
trueWinners.Select(w => resultsSets[w].SetName)));

    }

    var results = new List<Tuple<string, DataTable>>();
    results.Add(Tuple.Create("Comparison Grades",
comparisonGrades.AsDataTable()));
    results.Add(Tuple.Create("Comparison Results",
comparisonResults.AsDataTable()));
    if(interestingResults.Data != null){
      results.Add(Tuple.Create("Interesting Results",
interestingResults.AsDataTable()));
      results.Add(Tuple.Create("Interesting Results Std Dev",
interestingResultsStdev.AsDataTable()));
    }
    return results;
  }

  //Compares within a data set based on the test list passed, finds all
the columns that contain each test and compares them to each other
  public static List<Tuple<string, DataTable>>
DoInnerCompare(OneDimensionalDataSet<double?> resultsSet,
IEnumerable<string> testList){

    var stdDev = new TwoDimensionalDataSet<double>();
    var comparisonResults = new OneDimensionalDataSet<DataTable>();
    var comparisonGrades = new TwoDimensionalDataSet<object>();
    var interestingResults = new TwoDimensionalDataSet<object>();
    var interestingResultsStdev = new TwoDimensionalDataSet<object>();

    foreach(var test in testList.Distinct()){
```

```
      var columnsToCompare = new
List<string>(resultsSet.Data.Keys).Where(x =>
x.Contains(test)).ToList();
      if(columnsToCompare.Count <= 1){
        string.Format("Test {0} does not have enough items for
comparison. Items: {1}", test, string.Join(", ",
columnsToCompare)).Dump();
        continue;
      }
      List<List<int>> winners = new List<List<int>>();
      var allMeans = new Dictionary<string, double>();

      //compare each item with each other so if A, B and C then we test
A vs B, A vs C, and B vs C
      //If we have A, B, C, and D then we test A vs B, A vs C, A vs D,
B vs C, B vs D, C vs D
      for(int i = 0; i < columnsToCompare.Count; i++){
        for(int j = i + 1; j < columnsToCompare.Count; j++){
          string name1 = GetName(columnsToCompare[i], test);
          string name2 = GetName(columnsToCompare[j], test);
          var comparisonResult =
DataCompare.CompareDataSets(string.Format("{0} - {1} vs {2}", test,
name1, name2),

resultsSet.GetItems(columnsToCompare[i]).Where(x =>
x.HasValue).Select(x => x.Value).ToList(),
                            name1,

resultsSet.GetItems(columnsToCompare[j]).Where(x =>
x.HasValue).Select(x => x.Value).ToList(),
                            name2);

          stdDev.SetItem(test, name1,
comparisonResult.SetComparison.GetItem(name1, "Standard Deviation"));
          stdDev.SetItem(test, name2,
comparisonResult.SetComparison.GetItem(name2, "Standard Deviation"));
          comparisonResults.AddItem(test,
comparisonResult.SetComparison.AsDataTable());
          comparisonResults.AddItem(test,
comparisonResult.TTestResults.AsDataTable());

          allMeans[columnsToCompare[i]] =
comparisonResult.SetComparison.GetItem(name1, "Mean");
          allMeans[columnsToCompare[j]] =
comparisonResult.SetComparison.GetItem(name2, "Mean");

          //Set this round's winner(s)
          if(comparisonResult.SetsAreDifferent) {//Not a Tie
            winners.Add(new List<int>() { allMeans[columnsToCompare[i]]
> allMeans[columnsToCompare[j]] ? i : j });
          } else { //Tie
            winners.Add(new List<int>() { i, j });
          }
        }
      }
```

```
      //In order to truly have a winner they must have won or tied
every round with the other items
      var winsOrTiesRequired = columnsToCompare.Count - 1;
      var trueWinners = winners.SelectMany(w => w).GroupBy(w =>
w).Where(g => g.Count() >= winsOrTiesRequired).Select(g =>
g.Key).ToList();

      var maxMean = new List<double>(allMeans.Values).Max();

      //Set is interesting if the grade is < 90% of the max and someone
lost
      var isInteresting = allMeans.Values.Any(v => v / maxMean < .9) &&
allMeans.Count != trueWinners.Count;

      //Build grade based on largest mean, grade = ((mean / maxMean) *
100)%
      //If set is interesting, add it to the interesting results
      foreach(var column in columnsToCompare){
        var setGrade = Math.Round((allMeans[column] / maxMean) * 100.0,
2) + "%";
        comparisonGrades.SetItem(GetName(column, test), test,
setGrade);
        if(isInteresting){
          var stdDevGrade = Math.Round((stdDev.GetItem(test,
GetName(column, test)) / maxMean) * 100.0, 2) + "%";
          //interestingResults.SetItem(GetName(column, test), test,
(Math.Truncate(allMeans[column] * 1000.0) / 1000.0) + " (" + setGrade +
")");
          interestingResults.SetItem(GetName(column, test), test,
setGrade);
          interestingResultsStdev.SetItem(GetName(column, test), test,
stdDevGrade);
        }
      }

      //Set the winner row for this test based on the found winners
      comparisonGrades.SetItem("Winner", test, string.Join(" And ",
trueWinners.Select(w => GetName(columnsToCompare[w], test))));

    }

    var results = new List<Tuple<string, DataTable>>();
    results.Add(Tuple.Create("Comparison Grades",
comparisonGrades.AsDataTable()));
    results.Add(Tuple.Create("Comparison Results",
comparisonResults.AsDataTable()));
    if(interestingResults.Data != null){
      results.Add(Tuple.Create("Interesting Results",
interestingResults.AsDataTable()));
      results.Add(Tuple.Create("Interesting Results Std Dev",
interestingResultsStdev.AsDataTable()));
    }
    return results;
  }
```

```
  public static ComparisonResult CompareDataSets(string comparisonName,
List<double> set1, string set1Name, List<double> set2, string set2Name,
bool removeTails = true){
    var test = new TTestResult(){
      TestName = comparisonName,
      SetOne = new TTestSet() { SetData = set1 },
      SetTwo = new TTestSet() { SetData = set2 }
    };

    if(removeTails){
      test.SetOne.RemoveTails();
      test.SetTwo.RemoveTails();
    }

    var table = new TwoDimensionalDataSet<double>();

    table.SetItem(set1Name, "Mean", test.SetOne.Mean);
    table.SetItem(set1Name, "Standard Deviation", test.SetOne.StdDev);
    table.SetItem(set1Name, "Min", test.SetOne.Min);
    table.SetItem(set1Name, "Max", test.SetOne.Max);
    table.SetItem(set1Name, "Samples", test.SetOne.SampleSize);
    table.SetItem(set2Name, "Mean", test.SetTwo.Mean);
    table.SetItem(set2Name, "Standard Deviation", test.SetTwo.StdDev);
    table.SetItem(set2Name, "Min", test.SetTwo.Min);
    table.SetItem(set2Name, "Max", test.SetTwo.Max);
    table.SetItem(set2Name, "Samples", test.SetTwo.SampleSize);

    var totalsTable = new TwoDimensionalDataSet<object>();

    //This is where R is run
    var rResult = test.RNETResult;

    //Microsoft's p value
    var mP = test.p;

    //p < .05 = significant difference
    var areDifferent = rResult == null ? false : (double) rResult["p
value"] <.05;
    totalsTable.SetItem("P Value", "-", rResult == null ? (double?)
null: (double?) rResult["p value"]);
    totalsTable.SetItem("Significant Difference?", "-", areDifferent);

    //P values can be a little off, but if they are off by .001 or more
then the result could be wrong
    if(rResult != null && Math.Abs(((double) rResult["p value"]) - mP)
>= .001){
      throw new Exception("The p values between the two frameworks do
not match, this could mean that one or the other is wrong.");
    }
    return new ComparisonResult() { SetComparison = table, TTestResults
= totalsTable, SetsAreDifferent = areDifferent};
  }
}

public static class PerformanceResultsParser{
  public static OneDimensionalDataSet<string> ParseFiles(string
SetName, List<FileMapping> fileMappings){
```

```
    return ParseFiles(SetName, fileMappings, (value) => { return value;
});
  }

  public static OneDimensionalDataSet<T> ParseFiles<T>(string SetName,
List<FileMapping> fileMappings, Func<string, T> mapping){
    var results = new OneDimensionalDataSet<T>() { SetName = SetName };
    //For each file that we found and mapped read the CSV data with
CsvReader
    foreach(var fromToItem in fileMappings)
    {
      var path = fromToItem.FilePath;
      if (System.IO.File.Exists(path))
      {
        using (StreamReader sr = new StreamReader(path))
        {
          using (var csv = new CsvHelper.CsvReader(sr))
          {
            csv.Configuration.HasHeaderRecord = true;
            List<string> myStringColumn= new List<string>();
            //Data rows are seperated by 8 additional rows of metadata
about the test run so skip 8 lines after each row
            int take = 120, skip = 8, totalCount = 0, index = 0;
            while (csv.Read())
            {
              if(index == 0){
                string
stringField=csv.GetField<string>(fromToItem.OriginalColumnName);
                results.AddItem(fromToItem.ResultingColumnName,
mapping(stringField));

                totalCount++;
                if(totalCount == take){
                  break;
                }
              }

              index++;
              //Reset index to tell the next iteration to read in the
next row
              if(skip == 0 || index == skip){
                index = 0;
              }
            }
          }
        }
      } else {
        ("File not found: " + path).Dump();
      }
    }
    return results;
  }
}

public static class NetworkResultsParser{
  public static double GetThroughput(string filePath){
```

```
    Regex networkParserRegex = new
Regex(".*Throughput\\(Mbps\\)=(?<throughput>\\d+.*?\\d+).*");

    string text = System.IO.File.ReadAllText(filePath);

    Match match = networkParserRegex.Match(text);

    return double.Parse(match.Groups["throughput"].Value);
  }


    public static List<System.IO.FileInfo>
GetTxtFilesInDirectoryTree(System.IO.DirectoryInfo root, bool
includeThisDirectory = false)
    {
        List<System.IO.FileInfo> files = new
List<System.IO.FileInfo>();

        // First, process all the files directly under this folder
    if(includeThisDirectory){
      files.AddRange(root.GetFiles("*.txt"));
    }

        if (files != null)
        {
            // Now find all the subdirectories under this directory.
            System.IO.DirectoryInfo[] subDirs = root.GetDirectories();

            foreach (System.IO.DirectoryInfo dirInfo in subDirs)
            {
                // Resursive call for each subdirectory.
                files.AddRange(GetTxtFilesInDirectoryTree(dirInfo,
true));
            }
        }
    return files;
    }
}

//POCO
public class FileMapping {
  public string SystemClass { get; set; }
  public string FilePath { get; set; }
  public string OriginalColumnName { get; set; }
  public string ResultingColumnName { get; set; }

  public FileMapping(string systemClass, string filePath, string
originalColumnName, string resultingColumnName){
    this.SystemClass = systemClass;
    this.FilePath = filePath;
    this.OriginalColumnName = originalColumnName;
    this.ResultingColumnName = resultingColumnName;
  }
}

public class TTestResult{
  public string TestName { get; set; }
```

```
  public TTestSet SetOne { get; set; }
  public TTestSet SetTwo { get; set; }

  public int DegreesOfFreedom {
    get
    {
      if(SetOne == null || SetTwo == null){
        return -1;
      }

      var s1Variance = this.SetOne.Variance;
      var s1Count = this.SetOne.SampleSize;
      var s2Variance = this.SetTwo.Variance;
      var s2Count = this.SetTwo.SampleSize;

      // ((v1 / count1) + (v2 / count2)) ^ 2
      var top = Math.Pow((s1Variance / s1Count) + (s2Variance /
s2Count), 2.0);

      // (((v1 / count1) ^ 2) / (count1 - 1)) + (((v2 / count2) ^ 2) /
(count2 - 1))
      var bottom = (Math.Pow((s1Variance/s1Count), 2.0) / (s1Count -
1)) + (Math.Pow((s2Variance/s2Count), 2.0) / (s2Count - 1));

      return (int) Math.Round(top / bottom, 0,
MidpointRounding.AwayFromZero);
    }
  }

  public double t {
    get
    {
      if(SetOne == null || SetTwo == null){
        throw new InvalidOperationException("Both sets must be
populated");
      }

      return (this.SetOne.Mean - this.SetTwo.Mean) /
Math.Sqrt((this.SetOne.Variance / this.SetOne.SampleSize) +
(this.SetTwo.Variance / this.SetTwo.SampleSize));
    }
  }

  public double p {
    get
    {
      return StatisticalAnalysis.StudentsDistribution(this.t,
this.DegreesOfFreedom, false);
    }
  }

  public Dictionary<string, object> RNETResult {
    get
    {
      var engine = RDotNet.REngine.GetInstance();
      RDotNet.NumericVector group1 =
engine.CreateNumericVector(this.SetOne.SetData);
```

```
      engine.SetSymbol("group1", group1);
      RDotNet.NumericVector group2 =
engine.CreateNumericVector(this.SetTwo.SetData);
      engine.SetSymbol("group2", group2);

      // Test difference of mean and get the P-value.
      try {
        //used invisible() because latest R version printed test
details to the console
        var testResult =
RDotNet.SymbolicExpressionExtension.AsList(engine.Evaluate("invisible(t
.test(group1, group2, conf.level = 0.95))"));
        Dictionary<string, object> results = new Dictionary<string,
object>();
        results.Add("t value",
testResult["statistic"].AsNumeric().First());
        results.Add("p value",
RDotNet.SymbolicExpressionExtension.AsNumeric(testResult["p.value"].AsN
umeric()).First());
        results.Add("Degrees Of Freedom",
RDotNet.SymbolicExpressionExtension.AsNumeric(testResult["parameter"].A
sNumeric()).First());
        results.Add("Mean - Set One",
testResult["estimate"].AsNumeric().ElementAt(0));
        results.Add("Mean - Set Two",
testResult["estimate"].AsNumeric().ElementAt(1));
        results.Add("conf.int - 1",
testResult["conf.int"].AsNumeric().ElementAt(0)); //I Have no idea what
these are supposed to represent
        results.Add("conf.int - 2",
testResult["conf.int"].AsNumeric().ElementAt(1)); //I Have no idea what
these are supposed to represent
        results.Add("null.value",
testResult["null.value"].AsNumeric().ElementAt(0));

results.Add("alternative",testResult["alternative"].AsCharacter().Eleme
ntAt(0));
        results.Add("method",
testResult["method"].AsCharacter().ElementAt(0));
        return results;
      } catch(Exception e){
        //Don't care about the constant data sets exception
        if(e.Message.Contains("data are essentially constant")){
          return null;
        } else {
          e.Dump();
          this.TestName.Dump();
          throw;
        }
      }

    }
  }

}

public class TTestSet{
```

```
  public List<double> SetData { get; set; }
  public int SampleSize  {
    get
    {
      return this.SetData == null ? -1 : this.SetData.Count;
    }
  }

  public double Min  {
    get
    {
      return this.SetData == null ? -1 : this.SetData.Min();
    }
  }

  public double Max  {
    get
    {
      return this.SetData == null ? -1 : this.SetData.Max();
    }
  }

  public double Sum  {
    get
    {
      return this.SetData == null ? -1 : this.SetData.Sum();
    }
  }

  public double Mean  {
    get
    {
      return this.SetData == null ? -1 : this.Sum / this.SampleSize;
    }
  }

  public double Variance {
    get
    {
      if(this.SetData == null){
        return -1;
      }

      double mean = this.Mean;

      double result = this.SetData.Sum(num => Math.Pow((num - mean),
2));

            return result / (this.SampleSize - 1); //Of a population so
sample - 1
    }
  }

  public double StdDev {
    get
    {
            return Math.Sqrt(this.Variance);
```

```
    }
  }

  public double RemovedAbove { get; set; }
  public double RemovedBelow { get; set; }

  public void RemoveTails(double removalLevel = 5)  {
    var sorted = this.SetData.OrderBy(s => s).ToList();
    var middle = sorted.Count / 2.0;

    //first quartile median = middle * .5, third quartile median =
middle * 1.5 so a quartile's median is really middle * (quartile# / 2)
    Func<int, double> GetQuartileMedian = (quartile) =>
          {
            //if median is between two numbers (a + b) / 2, otherwise
(a + a) / 2
            return ((sorted.ElementAt((int) Math.Floor(middle *
(quartile / 2))) + sorted.ElementAt((int) Math.Ceiling(middle *
(quartile / 2)))) / 2);
          };

    var firstQuartileMedian = GetQuartileMedian(1);
    var thirdQuartileMedian = GetQuartileMedian(3);

    //iqr is the Interquartile Range, increasing the removal level
increases the range of accepted numbers for the set
    var iqr = (thirdQuartileMedian - firstQuartileMedian) *
removalLevel;
    var top = thirdQuartileMedian + iqr;
    var bottom = firstQuartileMedian - iqr;

    List<double> itemsOutsideIQR = this.SetData.Where(value => value >
top || value < bottom).ToList();

    foreach(var item in itemsOutsideIQR){
      this.SetData.Remove(item);
    }

    this.RemovedAbove = top;
    this.RemovedBelow = bottom;
  }
}

public interface IDimensionalDataSet {
  string SetName { get; set; }
  DataTable AsDataTable();
}

public class TwoDimensionalDataSet<T> : IDimensionalDataSet{
  public string SetName { get; set; }
  public Dictionary<string, Dictionary<string, T>> Data { get; set; }

  public void SetItem(string row, string column, T value){
    if(Data == null){
      Data = new Dictionary<string, Dictionary<string, T>>();
    }
    if(!Data.ContainsKey(row)){
```

```
      Data[row] = new Dictionary<string, T>();
    }
    Data[row][column] = value;
  }

  public bool HasItem(string row, string column){
    return Data != null && Data.ContainsKey(row) &&
Data[row].ContainsKey(column);
  }

  public T GetItem(string row, string column){
    if(Data == null){
      throw new InvalidOperationException("No data has been added to
the set yet.");
    }
    if(!Data.ContainsKey(row)){
      throw new IndexOutOfRangeException(string.Format("No item exists
at index {0} - {1}", row, column));
    }
    return Data[row][column];
  }

  public DataTable AsDataTable(){
    if(Data == null){
      throw new InvalidOperationException("No data has been added to
the set yet.");
    }
    DataTable table = new DataTable(SetName ?? "");

    table.Columns.Add(new DataColumn("*", typeof(string)));
    foreach(var column in new List<Dictionary<string,
T>>(Data.Values).SelectMany(x => x.Keys).Distinct()){
      table.Columns.Add(new DataColumn(column, typeof(object))); //
Object because i want to be able to use nullable primatives
    }

    foreach(var row in Data.Keys){
      var dataRow = table.NewRow();
      dataRow["*"] = row;
      foreach(var key in Data[row].Keys){
        dataRow[key] = Data[row][key];
      }
      table.Rows.Add(dataRow);
    }
    return table;
  }
}

public class OneDimensionalDataSet<T> : IDimensionalDataSet {
  public string SetName { get; set; }
  public Dictionary<string, List<T>> Data { get; set; }

  public void AddItem(string column, T value){
    if(Data == null){
      Data = new Dictionary<string, List<T>>();
    }
    if(!Data.ContainsKey(column)){
```

```
      Data[column] = new List<T>();
    }
    Data[column].Add(value);
  }

  public bool HasItem(string column){
    if(Data == null){
      return false;
    }
    return Data.ContainsKey(column);
  }

  public List<T> GetItems(string column){
    if(Data == null){
      throw new InvalidOperationException("No data has been added to
the set yet.");
    }
    if(!Data.ContainsKey(column)){
      throw new IndexOutOfRangeException(string.Format("No item exists
at index {0}", column));
    }
    return Data[column];
  }

  public DataTable AsDataTable(){
    if(Data == null){
      throw new InvalidOperationException("No data has been added to
the set yet.");
    }
    DataTable table = new DataTable(SetName ?? "");

    foreach(var column in Data.Keys){
      table.Columns.Add(new DataColumn(column, typeof(object))); //
Object because i want to be able to use nullable primatives
    }

    List<DataRow> dataRows = new List<DataRow>();
    foreach(var column in Data.Keys){
      var dataRow = table.NewRow();
      var items = Data[column];
      for (int i = 0; i < items.Count; i++)
      {
        if(dataRows.Count <= i){
          dataRows.Add(table.NewRow());
        }
        dataRows.ElementAt(i)[column] = items.ElementAt(i);
      }
    }
    foreach (var dataRow in dataRows)
    {
      table.Rows.Add(dataRow);
    }
    return table;
  }
}
```

```
//Methods are decompiled from .NET framework
System.Web.UI.DataVisualization.Charting.StatisticFormula.TDistribution
//i do not take credit for these methods, I pulled them out because i
wanted direct access to the method without going through the charting
api
public class StatisticalAnalysis
{
    private static double GammLn(double n)
    {
        double[] numArray = new double[6]
      {
        76.1800917294715,
        -86.5053203294168,
        24.0140982408309,
        -1.23173957245015,
        0.00120865097386618,
        -5.395239384953E-06
      };
        if (n < 0.0)
            throw new ArgumentOutOfRangeException("n");
        double num1;
        double num2 = num1 = n;
        double d = num1 + 5.5;
        double num3 = d - (num1 + 0.5) * Math.Log(d);
        double num4 = 1.00000000019001;
        for (int index = 0; index <= 5; ++index)
            num4 += numArray[index] / ++num2;
        return -num3 + Math.Log(2.506628274631 * num4 / num1);
    }

    private static double BetaCF(double a, double b, double x)
    {
        int num1 = 100;
        double num2 = 3E-07;
        double num3 = 1E-30;
        double num4 = a + b;
        double num5 = a + 1.0;
        double num6 = a - 1.0;
        double num7 = 1.0;
        double num8 = 1.0 - num4 * x / num5;
        if (Math.Abs(num8) < num3)
            num8 = num3;
        double num9 = 1.0 / num8;
        double num10 = num9;
        int num11;
        for (num11 = 1; num11 <= num1; ++num11)
        {
            int num12 = 2 * num11;
            double num13 = (double)num11 * (b - (double)num11) * x /
((num6 + (double)num12) * (a + (double)num12));
            double num14 = 1.0 + num13 * num9;
            if (Math.Abs(num14) < num3)
                num14 = num3;
            double num15 = 1.0 + num13 / num7;
            if (Math.Abs(num15) < num3)
                num15 = num3;
            double num16 = 1.0 / num14;
```

```
            double num17 = num10 * (num16 * num15);
            double num18 = -(a + (double)num11) * (num4 +
(double)num11) * x / ((a + (double)num12) * (num5 + (double)num12));
            double num19 = 1.0 + num18 * num16;
            if (Math.Abs(num19) < num3)
                num19 = num3;
            num7 = 1.0 + num18 / num15;
            if (Math.Abs(num7) < num3)
                num7 = num3;
            num9 = 1.0 / num19;
            double num20 = num9 * num7;
            num10 = num17 * num20;
            if (Math.Abs(num20 - 1.0) < num2)
                break;
        }
        if (num11 > num1)
            throw new InvalidOperationException("Invalid Result");
        else
            return num10;
    }

    private static double BetaIncomplete(double a, double b, double x)
    {
        if (x < 0.0 || x > 1.0)
            throw new ArgumentOutOfRangeException("x");
        double num = x == 0.0 || x == 1.0 ? 0.0 : Math.Exp(GammLn(a +
b) - GammLn(a) - GammLn(b) + a * Math.Log(x) + b * Math.Log(1.0 - x));
        if (x < (a + 1.0) / (a + b + 2.0))
            return num * BetaCF(a, b, x) / a;
        else
            return 1.0 - num * BetaCF(b, a, 1.0 - x) / b;
    }

    public static double StudentsDistribution(double tValue, int
degreesOfFreedom, bool oneTailed)
    {
        tValue = Math.Abs(tValue);
        if (degreesOfFreedom > 300)
            degreesOfFreedom = 300;
        if (degreesOfFreedom < 1)
            throw new ArgumentOutOfRangeException("degreesOfFreedom");
        double num = 1.0 - BetaIncomplete((double)degreesOfFreedom /
2.0, 0.5, (double)degreesOfFreedom / ((double)degreesOfFreedom + tValue
* tValue));
        if (oneTailed)
            return (1.0 - num) / 2.0;
        else
            return 1.0 - num;
    }

}
```

APPENDIX G

BATCH SCRIPTS

1) Paravirtual Network Testing Script

```
@echo off
setlocal EnableDelayedExpansion

set user=Administrator
set password=***OMITTED***

set host1=192.168.1.123
set host2=192.168.1.124
set host3=192.168.1.125
set host4=192.168.1.126
set host5=192.168.1.127
set host6=192.168.1.128

set local=192.168.1.129
set fileSender="C:\Program Files (x86)\Microsoft Corporation\NT
Testing TCP Tool\NTttcps.exe"
set fileReceiver="C:\Program Files (x86)\Microsoft Corporation\NT
Testing TCP Tool\NTttcpr.exe"
set filePrefix=C:\Results\network_results_remote_bridge_para
REM 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 26 28 29 30
set list=0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 26 28 29 30


NET USE U: "\\%host1%\Results" %password% /USER:Self\%user%
NET USE V: "\\%host2%\Results" %password% /USER:Self\%user%
NET USE W: "\\%host3%\Results" %password% /USER:Self\%user%
NET USE X: "\\%host4%\Results" %password% /USER:Self\%user%
NET USE Y: "\\%host5%\Results" %password% /USER:Self\%user%
NET USE Z: "\\%host6%\Results" %password% /USER:Self\%user%

xcopy U:\network_results* "C:\Users\Jarga\Dropbox\Thesis\Network
Tests\VMWare - Sender" /i /d /q /y /c
xcopy V:\network_results* "C:\Users\Jarga\Dropbox\Thesis\Network
Tests\VMWare - Receiver" /i /d /q /y /c
xcopy W:\network_results* "C:\Users\Jarga\Dropbox\Thesis\Network
Tests\Xen - Sender" /i /d /q /y /c
xcopy X:\network_results* "C:\Users\Jarga\Dropbox\Thesis\Network
Tests\Xen - Receiver" /i /d /q /y /c
xcopy Y:\network_results* "C:\Users\Jarga\Dropbox\Thesis\Network
Tests\KVM - Sender" /i /d /q /y /c
xcopy Z:\network_results* "C:\Users\Jarga\Dropbox\Thesis\Network
Tests\KVM - Receiver" /i /d /q /y /c
```

```
NET USE /DELETE U:
NET USE /DELETE V:
NET USE /DELETE W:
NET USE /DELETE X:
NET USE /DELETE Y:
NET USE /DELETE Z:
pause

REM VMWare
(for %%a in (%list%) do (
   taskkill /s %host1% /u Self\%user% /p %password% /IM NTttcps.exe
/f
   taskkill /s %host2% /u Self\%user% /p %password% /IM NTttcpr.exe
/f

   echo ""%fileReceiver% -m 1,0,%host1% 1,1,%host1% -a 8 -fr -f
"%filePrefix%_%%a.txt"""
   echo ""%fileSender% -m 1,0,%host2% 1,1,%host2% -a 6 -f
"%filePrefix%_%%a.txt"""

   REM pause

   psexec.exe \\%host2% -i -d -u Self\%user% -p %password% -
accepteula %fileReceiver% -m 1,0,%host1% 1,1,%host1% -a 8 -fr -f
"%filePrefix%_%%a.txt"
   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula %fileSender% -m 1,0,%host2% 1,1,%host2% -a 6 -f
"%filePrefix%_%%a.txt"

   REM Wait 30 seconds
   PING 1.1.1.1 -n 1 -w 30000 >NUL
   REM pause
))

REM Wait 2 minutes
PING 1.1.1.1 -n 1 -w 120000 >NUL

REM XEN
(for %%a in (%list%) do (
   taskkill /s %host3% /u Self\%user% /p %password% /IM NTttcps.exe
/f
   taskkill /s %host4% /u Self\%user% /p %password% /IM NTttcpr.exe
/f

   psexec.exe \\%host4% -i -d -u Self\%user% -p %password% -
accepteula %fileReceiver% -m 1,0,%host3% 1,1,%host3% -a 8 -fr -f
"%filePrefix%_%%a.txt"
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula %fileSender% -m 1,0,%host4% 1,1,%host4% -a 6 -f
"%filePrefix%_%%a.txt"

   REM Wait 30 seconds
   PING 1.1.1.1 -n 1 -w 30000 >NUL
   REM pause
))
```

```
REM Wait 2 minutes
PING 1.1.1.1 -n 1 -w 120000 >NUL

REM KVM
(for %%a in (%list%) do (
   taskkill /s %host5% /u Self\%user% /p %password% /IM NTttcps.exe
/f
   taskkill /s %host6% /u Self\%user% /p %password% /IM NTttcpr.exe
/f

   psexec.exe \\%host6% -i -d -u Self\%user% -p %password% -
accepteula %fileReceiver% -m 1,0,%host5% 1,1,%host5% -a 8 -fr -f
"%filePrefix%_%%a.txt"
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula %fileSender% -m 1,0,%host6% 1,1,%host6% -a 6 -f
"%filePrefix%_%%a.txt"

   REM Wait 30 seconds
   PING 1.1.1.1 -n 1 -w 30000 >NUL
   REM pause
))

pause
```

2) E1000 and No modifications Testing Script

```
@echo off
setlocal EnableDelayedExpansion

set user=Administrator
set password=***OMITTED***

set host1=192.168.1.123
set host2=192.168.1.124
set host3=192.168.1.125
set host4=192.168.1.126
set host5=192.168.1.127
set host6=192.168.1.128

set local=192.168.1.129
set fileSender="C:\Program Files (x86)\Microsoft Corporation\NT
Testing TCP Tool\NTttcps.exe"
set fileReceiver="C:\Program Files (x86)\Microsoft Corporation\NT
Testing TCP Tool\NTttcpr.exe"
set filePrefix=C:\Results\network_results_remote_bridge_e1000
REM 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 26 28 29 30
set list=0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 26 28 29 30


NET USE U: "\\%host1%\Results" %password% /USER:Self\%user%
NET USE V: "\\%host2%\Results" %password% /USER:Self\%user%
NET USE W: "\\%host3%\Results" %password% /USER:Self\%user%
NET USE X: "\\%host4%\Results" %password% /USER:Self\%user%
```

```
NET USE Y: "\\%host5%\Results" %password% /USER:Self\%user%
NET USE Z: "\\%host6%\Results" %password% /USER:Self\%user%

xcopy U:\network_results_remote_bridge*
"C:\Users\Jarga\Dropbox\Thesis\Network Tests\VMWare - Sender" /i /d
/q /y /c
xcopy V:\network_results_remote_bridge*
"C:\Users\Jarga\Dropbox\Thesis\Network Tests\VMWare - Receiver" /i
/d /q /y /c
xcopy W:\network_results_remote_bridge*
"C:\Users\Jarga\Dropbox\Thesis\Network Tests\Xen - Sender" /i /d /q
/y /c
xcopy X:\network_results_remote_bridge*
"C:\Users\Jarga\Dropbox\Thesis\Network Tests\Xen - Receiver" /i /d
/q /y /c
xcopy Y:\network_results_remote_bridge*
"C:\Users\Jarga\Dropbox\Thesis\Network Tests\KVM - Sender" /i /d /q
/y /c
xcopy Z:\network_results_remote_bridge*
"C:\Users\Jarga\Dropbox\Thesis\Network Tests\KVM - Receiver" /i /d
/q /y /c

NET USE /DELETE U:
NET USE /DELETE V:
NET USE /DELETE W:
NET USE /DELETE X:
NET USE /DELETE Y:
NET USE /DELETE Z:
pause

:VMWare
(for %%a in (%list%) do (
   taskkill /s %host1% /u Self\%user% /p %password% /IM NTttcps.exe
/f
   taskkill /s %host2% /u Self\%user% /p %password% /IM NTttcpr.exe
/f

   echo ""%fileReceiver% -m 1,0,%host1% 1,1,%host1% -a 8 -fr -f
"%filePrefix%_%%a.txt"""
   echo ""%fileSender% -m 1,0,%host2% 1,1,%host2% -a 6 -f
"%filePrefix%_%%a.txt"""

   REM pause

   psexec.exe \\%host2% -i -d -u Self\%user% -p %password% -
accepteula %fileReceiver% -m 1,0,%host1% 1,1,%host1% -a 8 -fr -f
"%filePrefix%_%%a.txt"
   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula %fileSender% -m 1,0,%host2% 1,1,%host2% -a 6 -f
"%filePrefix%_%%a.txt"

   REM Wait 60 seconds
   PING 1.1.1.1 -n 1 -w 60000 >NUL
   REM pause
))

REM Wait 2 minutes
```

```
PING 1.1.1.1 -n 1 -w 120000 >NUL

:XEN
(for %%a in (%list%) do (
   taskkill /s %host3% /u Self\%user% /p %password% /IM NTttcps.exe
/f
   taskkill /s %host4% /u Self\%user% /p %password% /IM NTttcpr.exe
/f

   psexec.exe \\%host4% -i -d -u Self\%user% -p %password% -
accepteula %fileReceiver% -m 1,0,%host3% 1,1,%host3% -a 8 -fr -f
"%filePrefix%_%%a.txt"
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula %fileSender% -m 1,0,%host4% 1,1,%host4% -a 6 -f
"%filePrefix%_%%a.txt"

   REM Wait 30 seconds
   PING 1.1.1.1 -n 1 -w 120000 >NUL
   REM pause
))

REM Wait 2 minutes
PING 1.1.1.1 -n 1 -w 120000 >NUL

:KVM
REM KVM
(for %%a in (%list%) do (
   taskkill /s %host5% /u Self\%user% /p %password% /IM NTttcps.exe
/f
   taskkill /s %host6% /u Self\%user% /p %password% /IM NTttcpr.exe
/f

   psexec.exe \\%host6% -i -d -u Self\%user% -p %password% -
accepteula %fileReceiver% -m 1,0,%host5% 1,1,%host5% -a 8 -fr -f
"%filePrefix%_%%a.txt"
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula %fileSender% -m 1,0,%host6% 1,1,%host6% -a 6 -f
"%filePrefix%_%%a.txt"

   REM Wait 30 seconds
   PING 1.1.1.1 -n 1 -w 360000 >NUL
   REM pause
))

:END
pause
```

3) Paravirtual Parallel Testing Script

```
@echo off
setlocal EnableDelayedExpansion

set user=Administrator
set password=***OMITTED***
```

```
set host1=192.168.1.123
set host2=192.168.1.124
set host3=192.168.1.125
set host4=192.168.1.126
set host5=192.168.1.127
set host6=192.168.1.128

set remoteScript="C:\Program
Files\PerformanceTest\PerfTests.ptscript"

set local=192.168.1.129
set
file="C:\Users\Administrator\Desktop\ParallelExecution\ParallelExecu
tion.exe"

set localCmd="ParallelExecution.exe" -f "C:\Program
Files\PerformanceTest\PerformanceTest64.exe" -t 300000 -a "/s
"%remoteScript%" /NO3D" -r
"%host1%,%host2%,%host3%,%host4%,%host5%,%host6%"

set list=CPU_INTEGERMATH CPU_FLOATINGPOINTMATH CPU_PRIME CPU_SSE
CPU_COMPRESSION CPU_ENCRYPTION CPU_PHYSICS CPU_SORTING
CPU_SINGLETHREAD DI_WRITE DI_READ DI_RANDOM ME_ALLOC_S
ME_READ_CACHED ME_READ_UNCACHED ME_WRITE ME_LARGE ME_LATENCY
ME_THREADED


NET USE U: "\\%host1%\Results" %password% /USER:Self\%user%
NET USE V: "\\%host2%\Results" %password% /USER:Self\%user%
NET USE W: "\\%host3%\Results" %password% /USER:Self\%user%
NET USE X: "\\%host4%\Results" %password% /USER:Self\%user%
NET USE Y: "\\%host5%\Results" %password% /USER:Self\%user%
NET USE Z: "\\%host6%\Results" %password% /USER:Self\%user%

xcopy U:\*results-multi* "C:\Users\Jarga\Dropbox\Thesis\Parallel
Perf Tests\Results-VMWare-1" /i /d /q /y /c
xcopy V:\*results-multi* "C:\Users\Jarga\Dropbox\Thesis\Parallel
Perf Tests\Results-VMWare-2" /i /d /q /y /c
xcopy W:\*results-multi* "C:\Users\Jarga\Dropbox\Thesis\Parallel
Perf Tests\Results-XEN-1" /i /d /q /y /c
xcopy X:\*results-multi* "C:\Users\Jarga\Dropbox\Thesis\Parallel
Perf Tests\Results-XEN-2" /i /d /q /y /c
xcopy Y:\*results-multi* "C:\Users\Jarga\Dropbox\Thesis\Parallel
Perf Tests\Results-KVM-1" /i /d /q /y /c
xcopy Z:\*results-multi* "C:\Users\Jarga\Dropbox\Thesis\Parallel
Perf Tests\Results-KVM-2" /i /d /q /y /c

NET USE /DELETE U:
NET USE /DELETE V:
NET USE /DELETE W:
NET USE /DELETE X:
NET USE /DELETE Y:
NET USE /DELETE Z:
pause

set setScript=""
(for %%a in (%list%) do (
```

```
   echo "Starting: " %%a
   set setScript="( echo LOOP 30 & echo { & echo CLEARRESULTS & echo
RUN %%a & echo } REPORTSUMMARYCSV "C:\Results\results-multi-%%a.csv"
) ^> %remoteScript%"
   echo "Set script: " !setScript!

   REM Kill perf test windows and prev running jobs
   taskkill /s %host1% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host1% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host2% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host2% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host3% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host3% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host4% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host4% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host5% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host5% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host6% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host6% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f

   REM Set script contents
   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host2% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host4% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host6% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!

   REM Start local server
   start cmd.exe /k "%localCmd%"

   REM echo "%file% -s "!local!""

   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host2% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
```

```
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host4% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host6% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"


   REM Wait 40 minutes
   PING 1.1.1.1 -n 1 -w 2400000 >NUL
   REM pause

   REM Kill local server
   taskkill /IM ParallelExecution.exe /f
))

pause
```

4) No Modifications Parallel Testing Script

```
@echo off
setlocal EnableDelayedExpansion

set user=Administrator
set password=***OMITTED***

set host1=192.168.1.123
set host2=192.168.1.124
set host3=192.168.1.125
set host4=192.168.1.126
set host5=192.168.1.127
set host6=192.168.1.128

set remoteScript="C:\Program
Files\PerformanceTest\PerfTests.ptscript"

set local=192.168.1.129
set
file="C:\Users\Administrator\Desktop\ParallelExecution\ParallelExecu
tion.exe"

set localCmd="ParallelExecution.exe" -f "C:\Program
Files\PerformanceTest\PerformanceTest64.exe" -t 300000 -a "/s
"%remoteScript%" /NO3D" -r
"%host1%,%host2%,%host3%,%host4%,%host5%,%host6%"

set list=CPU_INTEGERMATH CPU_FLOATINGPOINTMATH CPU_PRIME CPU_SSE
CPU_COMPRESSION CPU_ENCRYPTION CPU_PHYSICS CPU_SORTING
CPU_SINGLETHREAD DI_WRITE DI_READ DI_RANDOM ME_ALLOC_S
ME_READ_CACHED ME_READ_UNCACHED ME_WRITE ME_LARGE ME_LATENCY
ME_THREADED
```

```
NET USE U: "\\%host1%\Results" %password% /USER:Self\%user%
NET USE V: "\\%host2%\Results" %password% /USER:Self\%user%
NET USE W: "\\%host3%\Results" %password% /USER:Self\%user%
NET USE X: "\\%host4%\Results" %password% /USER:Self\%user%
NET USE Y: "\\%host5%\Results" %password% /USER:Self\%user%
NET USE Z: "\\%host6%\Results" %password% /USER:Self\%user%

xcopy U:\nomod-results-multi*
"C:\Users\Jarga\Dropbox\Thesis\Parallel Perf Tests\NoMod-Results-
VMWare-1" /i /d /q /y /c
xcopy V:\nomod-results-multi*
"C:\Users\Jarga\Dropbox\Thesis\Parallel Perf Tests\NoMod-Results-
VMWare-2" /i /d /q /y /c
xcopy W:\nomod-results-multi*
"C:\Users\Jarga\Dropbox\Thesis\Parallel Perf Tests\NoMod-Results-
XEN-1" /i /d /q /y /c
xcopy X:\nomod-results-multi*
"C:\Users\Jarga\Dropbox\Thesis\Parallel Perf Tests\NoMod-Results-
XEN-2" /i /d /q /y /c
xcopy Y:\nomod-results-multi*
"C:\Users\Jarga\Dropbox\Thesis\Parallel Perf Tests\NativeIOMode-
Results-KVM-1" /i /d /q /y /c
xcopy Z:\nomod-results-multi*
"C:\Users\Jarga\Dropbox\Thesis\Parallel Perf Tests\NativeIOMode-
Results-KVM-2" /i /d /q /y /c

NET USE /DELETE U:
NET USE /DELETE V:
NET USE /DELETE W:
NET USE /DELETE X:
NET USE /DELETE Y:
NET USE /DELETE Z:
pause

set setScript=""
(for %%a in (%list%) do (

   echo "Starting: " %%a
   set setScript="( echo LOOP 30 & echo { & echo CLEARRESULTS & echo
RUN %%a & echo } REPORTSUMMARYCSV "C:\Results\nomod-results-multi-
%%a.csv" ) ^> %remoteScript%"
   echo "Set script: " !setScript!

   REM Kill perf test windows and prev running jobs
   taskkill /s %host1% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host1% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host2% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host2% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host3% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host3% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
```

```
   taskkill /s %host4% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host4% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host5% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host5% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f
   taskkill /s %host6% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host6% /u Self\%user% /p %password% /IM
ParallelExecution.exe /f

   REM Set script contents
   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host2% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host4% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host6% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!

   REM Start local server
   start cmd.exe /k "%localCmd%"

   REM echo "%file% -s "!local!""

   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host2% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host4% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"
   psexec.exe \\%host6% -i -d -u Self\%user% -p %password% -
accepteula %file% -s "%local%"


   REM Wait 40 minutes
   PING 1.1.1.1 -n 1 -w 2400000 >NUL
   REM pause

   REM Kill local server
   taskkill /IM ParallelExecution.exe /f
))

pause
```

5) Paravirtual Single Testing Script

```
@echo off
setlocal EnableDelayedExpansion

set user=Administrator
set password=***OMITTED***

set host1=192.168.1.123
set host2=192.168.1.124
set host3=192.168.1.125
set host4=192.168.1.126
set host5=192.168.1.127
set host6=192.168.1.128

set remoteScript="C:\Program
Files\PerformanceTest\PerfTests.ptscript"

set local=192.168.1.129
set file="C:\Program Files\PerformanceTest\PerformanceTest64.exe"

REM CPU_INTEGERMATH CPU_FLOATINGPOINTMATH CPU_PRIME CPU_SSE
CPU_COMPRESSION CPU_ENCRYPTION CPU_PHYSICS CPU_SORTING
CPU_SINGLETHREAD DI_WRITE DI_READ DI_RANDOM ME_ALLOC_S
ME_READ_CACHED ME_READ_UNCACHED ME_WRITE ME_LARGE ME_LATENCY
ME_THREADED
set list=CPU_INTEGERMATH CPU_FLOATINGPOINTMATH CPU_PRIME CPU_SSE
CPU_COMPRESSION CPU_ENCRYPTION CPU_PHYSICS CPU_SORTING
CPU_SINGLETHREAD DI_WRITE DI_READ DI_RANDOM ME_ALLOC_S
ME_READ_CACHED ME_READ_UNCACHED ME_WRITE ME_LARGE ME_LATENCY
ME_THREADED


NET USE U: "\\%host1%\Results" %password% /USER:Self\%user%
NET USE W: "\\%host3%\Results" %password% /USER:Self\%user%
NET USE Y: "\\%host5%\Results" %password% /USER:Self\%user%

xcopy U:\results-single* "C:\Users\Jarga\Dropbox\Thesis\Perf
Tests\Results-VMWare-1" /i /d /q /y /c
xcopy W:\results-single* "C:\Users\Jarga\Dropbox\Thesis\Perf
Tests\Results-XEN-1" /i /d /q /y /c
xcopy Y:\results-single* "C:\Users\Jarga\Dropbox\Thesis\Perf
Tests\Results-KVM-1" /i /d /q /y /c

NET USE /DELETE U:
NET USE /DELETE W:
NET USE /DELETE Y:
pause

set setScript=""
(for %%a in (%list%) do (

   REM for /f "usebackq tokens=2 delims=:" %%f in (`ipconfig ^|
findstr /c:"IPv4 Address"`) do set local=%%f
   REM set local = !local:~1!
   REM echo "Local IP: " !local!
```

```
   echo "Starting: " %%a
   set setScript="( echo LOOP 30 & echo { & echo CLEARRESULTS & echo
RUN %%a & echo } REPORTSUMMARYCSV "C:\Results\results-single-
%%a.csv" ) ^> %remoteScript%"
   echo "Set script: " !setScript!

   REM Kill perf test windows and prev running jobs
   taskkill /s %host1% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host3% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host5% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f

   echo "%file% /s "%remoteScript%" /NO3D"

   REM Set script contents
   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!

   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula %file% /s %remoteScript% /NO3D
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula %file% /s %remoteScript% /NO3D
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula %file% /s %remoteScript% /NO3D


   REM Wait 40 minutes
   PING 1.1.1.1 -n 1 -w 2400000 >NUL
   REM pause
))

pause
```

6) No Modifications Single Testing Script

```
@echo off
setlocal EnableDelayedExpansion

set user=Administrator
set password=***OMITTED***

set host1=192.168.1.123
set host2=192.168.1.124
set host3=192.168.1.125
set host4=192.168.1.126
set host5=192.168.1.127
set host6=192.168.1.128
```

```
set remoteScript="C:\Program
Files\PerformanceTest\PerfTests.ptscript"

set local=192.168.1.129
set file="C:\Program Files\PerformanceTest\PerformanceTest64.exe"

REM CPU_INTEGERMATH CPU_FLOATINGPOINTMATH CPU_PRIME CPU_SSE
CPU_COMPRESSION CPU_ENCRYPTION CPU_PHYSICS CPU_SORTING
CPU_SINGLETHREAD DI_WRITE DI_READ DI_RANDOM ME_ALLOC_S
ME_READ_CACHED ME_READ_UNCACHED ME_WRITE ME_LARGE ME_LATENCY
ME_THREADED
set list=CPU_INTEGERMATH CPU_FLOATINGPOINTMATH CPU_PRIME CPU_SSE
CPU_COMPRESSION CPU_ENCRYPTION CPU_PHYSICS CPU_SORTING
CPU_SINGLETHREAD DI_WRITE DI_READ DI_RANDOM ME_ALLOC_S
ME_READ_CACHED ME_READ_UNCACHED ME_WRITE ME_LARGE ME_LATENCY
ME_THREADED


NET USE U: "\\%host1%\Results" %password% /USER:Self\%user%
NET USE W: "\\%host3%\Results" %password% /USER:Self\%user%
NET USE Y: "\\%host5%\Results" %password% /USER:Self\%user%

xcopy U:\nomods-results-single* "C:\Users\Jarga\Dropbox\Thesis\Perf
Tests\NoMods-Results-VMWare-1" /i /d /q /y /c
xcopy W:\nomods-results-single* "C:\Users\Jarga\Dropbox\Thesis\Perf
Tests\NoMods-Results-XEN-1" /i /d /q /y /c
xcopy Y:\nomods-results-single* "C:\Users\Jarga\Dropbox\Thesis\Perf
Tests\NoMods-Results-KVM-1" /i /d /q /y /c

NET USE /DELETE U:
NET USE /DELETE W:
NET USE /DELETE Y:
pause

set setScript=""
(for %%a in (%list%) do (

   REM for /f "usebackq tokens=2 delims=:" %%f in (`ipconfig ^|
findstr /c:"IPv4 Address"`) do set local=%%f
   REM set local = !local:~1!
   REM echo "Local IP: " !local!

   echo "Starting: " %%a
   set setScript="( echo LOOP 30 & echo { & echo CLEARRESULTS & echo
RUN %%a & echo } REPORTSUMMARYCSV "C:\Results\nomods-results-single-
%%a.csv" ) ^> %remoteScript%"
   echo "Set script: " !setScript!

   REM Kill perf test windows and prev running jobs
   taskkill /s %host1% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host3% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f
   taskkill /s %host5% /u Self\%user% /p %password% /IM
PerformanceTest64.exe /f

   echo "%file% /s "%remoteScript%" /NO3D"
```

```
   REM Set script contents
   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula cmd.exe /c !setScript!

   psexec.exe \\%host1% -i -d -u Self\%user% -p %password% -
accepteula %file% /s %remoteScript% /NO3D
   psexec.exe \\%host3% -i -d -u Self\%user% -p %password% -
accepteula %file% /s %remoteScript% /NO3D
   psexec.exe \\%host5% -i -d -u Self\%user% -p %password% -
accepteula %file% /s %remoteScript% /NO3D


   REM Wait 40 minutes
   PING 1.1.1.1 -n 1 -w 2400000 >NUL
   REM pause
))

pause
```

APPENDIX H

PARALLEL EXECUTION V1.0

1) Options.cs

```
using CommandLine;
using CommandLine.Text;
using System.Collections.Generic;
using System.Text;

namespace ParallelExecution
{
    public class Options
    {
        [Option(shortName: 's', longName: "server", HelpText = "Server
to use as the controller of the processes when running as a client.")]
        public string Server { get; set; }

        [Option(shortName: 'p', longName: "port", HelpText = "Port to
connect to when running as a client or port to activly listen to
connections on if running as the server.", DefaultValue = 8888)]
        public int Port { get; set; }

        [Option(shortName: 't', longName: "timeout", HelpText =
"Timeout on for the server to stop accepting connections or the client
to stop waiting for a command (in milliseconds).", DefaultValue =
180000)]
        public int Timeout { get; set; }

        [Option(shortName: 'u', longName: "useserver", HelpText = "Use
this flag to execute the file on the server as well.")]
        public bool UseServer { get; set; }

        [OptionList('r', "receivers", Separator = ',', HelpText = "All
the receiving clients that will be used to execute the processes,
separated by a comma.")]
        public IList<string> Clients { get; set; }

        [Option(shortName: 'f', longName: "fileName", HelpText = "File
to execute.")]
        public string File { get; set; }

        [Option(shortName: 'a', longName: "arguments", HelpText =
"Arguments to pass to the file.")]
        public string Arguments { get; set; }

        [HelpOption]
        public string GetUsage()
        {
```

```
            var help = new HelpText
            {
                Heading = new HeadingInfo("\r\nParallel Execution
Application", "1.0"),
                Copyright = new CopyrightInfo("Sean McAdams", 2014),
                AdditionalNewLineAfterOption = true,
                AddDashesToOption = true,
            };
            help.AddPreOptionsLine("\r\nExample Usage:
ParallelExecution.exe -u -f notepad.exe -r \"172.0.0.0\"");
            help.AddPreOptionsLine(    "        Usage:
ParallelExecution.exe -s \"172.0.0.0\"");
            help.AddOptions(this);
            return help;
        }
    }
}
```

2) CommandExecutionClient.cs

```
using System;
using System.Net.Sockets;
using System.IO;
using System.Diagnostics;

namespace ParallelExecution
{
    public class CommandExecutionClient
    {
        private int _port;
        private string _server;
        private int _timeout;

        public CommandExecutionClient(int port, string server, int
timeout)
        {
            this._port = port;
            this._server = server;
            this._timeout = timeout;
        }

        public Process Start()
        {
            using (var client = new TcpClient(_server.Trim(), _port))
            {
                var stream = client.GetStream();

                using (var writer = new StreamWriter(stream))
                using (var reader = new StreamReader(stream))
                {
                    reader.BaseStream.ReadTimeout = this._timeout;
                    var command = reader.ReadLine();
```

```
                    if (command == null)
                    {
                        Console.WriteLine("Failed to read TCP
Stream!!");

                        return null;
                    }

                    var index =
command.IndexOf(ParallelExecutionMaster.CommandDelimiter);
                    if (index > -1)
                    {
                        var fileName = command.Substring(0, index);
                        var arguments = command.Substring(index +
ParallelExecutionMaster.CommandDelimiter.Length);

                        Console.WriteLine("Executing command: {0} {1}
at {2}", fileName, arguments, DateTime.Now.ToString("O"));

                        Process process = null;

                        if (string.IsNullOrWhiteSpace(arguments))
                        {
                            process = Process.Start(fileName);
                        }
                        else
                        {
                            process = Process.Start(fileName,
arguments);
                        }

                        writer.WriteLine("Success");
                        writer.Flush();

                        return process;
                    }

                    Console.WriteLine("Invalid Format Command Format
Sent to Client!");

                    writer.WriteLine("Failed");
                }
            }
            return null;
        }
    }
}
```

3) CommandExecutionServer.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Threading.Tasks;
using System.Diagnostics;
using System.IO;

namespace ParallelExecution
{
    public class CommandExecutionServer
    {
        private IList<string> _clients;
        private bool _runOnServer = false;
        private int _port;
        private int _serverTimeout;
        private string _fileName;
        private string _arguments;
        private IList<string> _clientRemainingToConnect = new
List<string>();
        private IList<TcpClient> _connectedClients = new
List<TcpClient>();

        public CommandExecutionServer(string fileName, string
arguments, IList<string> clients)
        {
            _fileName = fileName;
            _arguments = arguments;
            _clients = clients;
            _clientRemainingToConnect = _clients;
        }

        public CommandExecutionServer(int port, string fileName, string
arguments, IList<string> clients)
        {
            _fileName = fileName;
            _arguments = arguments;
            _clients = clients;
            _clientRemainingToConnect = _clients;
            _port = port;
        }

        public CommandExecutionServer(int port, int serverTimeout,
string fileName, string arguments, IList<string> clients)
        {
            _fileName = fileName;
            _arguments = arguments;
            _clients = clients;
            _clientRemainingToConnect = _clients;
            _port = port;
            _serverTimeout = serverTimeout;
        }
```

```
        public CommandExecutionServer(int port, int serverTimeout,
string fileName, string arguments, IList<string> clients, bool
runOnServer)
        {
            _fileName = fileName;
            _arguments = arguments;
            _clients = clients;
            _clientRemainingToConnect = _clients;
            _port = port;
            _serverTimeout = serverTimeout;
            _runOnServer = runOnServer;
        }

        public bool Start()
        {
            var serverListener = new TcpListener(IPAddress.Any, _port);

            var watch = new Stopwatch();

            serverListener.Start();
            watch.Start();

            while (watch.ElapsedMilliseconds < _serverTimeout)
            {
                if (serverListener.Pending())
                {
                    var connection = serverListener.AcceptTcpClient();
                    var client = (connection.Client.RemoteEndPoint as
IPEndPoint).Address.ToString();
                    if (_clientRemainingToConnect.Contains(client))
                    {
                        _clientRemainingToConnect.Remove(client);
                        _connectedClients.Add(connection);
                    }
                    else
                    {
                        Console.WriteLine("Recieved unexpected
connection request from {0}, closing connection.", client);
                        connection.Close();
                    }
                }

                if (_clientRemainingToConnect.Count == 0)
                {
                    watch.Stop();
                    Console.WriteLine("Server waited {0} ms for clients
to connect!", watch.ElapsedMilliseconds);
                    return StartExecution();
                }

            }

            watch.Stop();
```

```
            Console.WriteLine("Server waited {0} ms for clients to
connect!", watch.ElapsedMilliseconds);

            return false;
        }

        private bool StartExecution()
        {
            Console.WriteLine("All client are now connected, sending
command to each client!");

            var tasks = new List<Task<bool>>();

            for (int i = 0; i < _connectedClients.Count; i++)
            {
                var connection = _connectedClients[i];
                tasks.Add(new Task<bool>(() =>
                        {
                            Console.WriteLine("Triggering command: {0}
{1} at {2} for Machine {3}", _fileName.Trim(), _arguments,
DateTime.Now.ToString("O"), (connection.Client.RemoteEndPoint as
IPEndPoint).Address.ToString());

                            var stream = connection.GetStream();

                            using (var writer = new
StreamWriter(stream))
                            using (var reader = new
StreamReader(stream))
                            {
                                writer.WriteLine("{0}{1}{2}",
_fileName.Trim(), ParallelExecutionMaster.CommandDelimiter,
_arguments);

                                writer.Flush();

                                var returnVal = reader.ReadLine();

                                return "Success".Equals(returnVal);
                            }

                        }
                ));
            }

            tasks.ForEach(task => task.Start());

            if (_runOnServer)
            {
                Console.WriteLine("Executing command: {0} {1} at {2}",
_fileName, _arguments, DateTime.Now.ToString("O"));

                if (string.IsNullOrWhiteSpace(_arguments))
                {
```

```
                    Process.Start(_fileName);
                }
                else
                {
                    Process.Start(_fileName, _arguments);
                }

            }

            tasks.ForEach(task => task.Wait());

            var result = tasks.All(task => task.Result);

            Console.WriteLine(result ? "Processes started!" : "At least
one of the clients failed to start the process!");

            return result;
        }
    }
}
```

4) ParallelExecutionMaster.cs

```
using CommandLine;
using System;
using System.Linq;

namespace ParallelExecution
{
    public class ParallelExecutionMaster
    {
        /// <summary>
        /// Delimiter to use between the filename and arguments when
sending the execution request from the server so the client can parse
it easier, needs to be an uncommon series of characters
        /// </summary>
        public const string CommandDelimiter = "||";

        static void Main(string[] args)
        {
            var options = new Options();

            var parser = new Parser();


            //If parsing was successful verify either a server is given
or the file and receivers are given
            if (parser.ParseArguments(args, options) &&
(!string.IsNullOrWhiteSpace(options.Server) ||
(!string.IsNullOrWhiteSpace(options.File) && options.Clients != null &&
options.Clients.Any())))
            {
```

```
                //If No Server is Given assume Current Box is the
Server otherwise process act as if you are a client
                if (string.IsNullOrWhiteSpace(options.Server))
                {
                    var server = new
CommandExecutionServer(options.Port, options.Timeout, options.File,
options.Arguments, options.Clients, options.UseServer);

                    var result = server.Start();

                    Console.WriteLine(result ? "Execution Successful!
Processes Running." : "An Error Occured while attempting to execute the
processes!");
                }
                else
                {
                    var client = new
CommandExecutionClient(options.Port, options.Server, options.Timeout);

                    var process = client.Start();

                    if (process != null)
                    {
                        Console.WriteLine("Process Started!");
                        process.WaitForExit();
                        Console.WriteLine("Process Complete!");
                    }
                    else
                    {
                        Console.WriteLine("Error occured when
attempting to execute process!");
                    }

                }
            }
            else
            {
                Console.WriteLine(options.GetUsage());
                return;
            }

            Console.WriteLine("Press Enter To Close The Console!");
            Console.ReadLine();
        }
    }
}
```

VITA

Sean McAdams has a Bachelor of Science in Computer and Information Sciences from the University of North Florida and expects to receive a Master of Science in Software Engineering from the University of North Florida, October 2015. Dr. Roger Eggen of the University of North Florida is serving as Sean McAdams' thesis advisor. Sean is a software developer currently employed by OceansideTen Management, LLC and is the owner of Sigma 8, LLC a contract software development firm. Sean has over 5 years of experience in software development and has worked in enterprise environments for some of the largest financial corporations in the world. His recent work mostly includes Microsoft based solutions using the .NET framework in the C# programming language and continually pushes himself to learn new technologies to expand his skills. Sean's goal is to expand his knowledge of software development and build his company into a successful software development firm.