2012

A Performance Comparison of Hypervisors for Cloud Computing

Suganya Sridharan
University of North Florida

A PERFORMANCE COMPARISON OF HYPERVISORS FOR CLOUD COMPUTING

By

Suganya Sridharan

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

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

August 2012

The thesis "A Performance Comparison of Hypervisors for Cloud Computing" submitted by Suganya Sridharan in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the thesis committee: Date

Signature Deleted

4/23/2012

Dr. Sanjay Ahuja
Thesis Advisor and Committee Chairperson

Signature Deleted

4/23/2012

Dr. Roger Eggen
Graduate Director and Professor

Signature Deleted

4/23/2012

Dr. Zornitza Prodanoff
Associate Professor of Computing

Accepted for the School of Computing:

Signature Deleted

6.11.2012

Dr. Asai Asaithambi
Director of the School

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

Signature Deleted

6/19/12

Dr. Mark A.Tumeo
Dean of the College

Accepted for the University:

Signature Deleted

8/6/12

Dr. Len Roberson
Dean of the Graduate School

ACKNOWLEDGEMENT

I wish to thank my parents, Sridharan Rangasamy and Suriya Sridharan, and my spouse, Raaja Raajan Angathevar Veluchamy, for their unwavering support and concern during the many hours I dedicated to achieve this milestone in my life and career. I highly appreciate the understanding and patience of my family for achieving one of the greatest moments of my lifetime.

I thank my thesis director Dr. Sanjay Ahuja for his expert advice, feedback and suggestions. I also thank Dr. Roger Eggen and Dr. Zornitza Prodanoff who agreed to be on my thesis committee and provided great feedback in this research process. Working with these three distinguished UNF faculty members has been an honor and a privilege. I also thank School of Computing for ordering SPECvirt_sc2010 standard benchmark required for this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The virtualization of IT infrastructure enables the consolidation and pooling of IT resources so that they can be shared over diverse applications to offset the limitation of shrinking resources and growing business needs. Virtualization provides a logical abstraction of physical computing resources and creates computing environments that are not restricted by physical configuration or implementation. Virtualization is very important for cloud computing because the delivery of services is simplified by providing a platform for optimizing complex IT resources in a scalable manner, which makes cloud computing more cost effective.

Hypervisor plays an important role in the virtualization of hardware. It is a piece of software that provides a virtualized hardware environment to support running multiple operating systems concurrently using one physical server. Cloud computing has to support multiple operating environments and Hypervisor is the ideal delivery mechanism.

The intent of this thesis is to quantitatively and qualitatively compare the performance of VMware ESXi 4.1, Citrix Systems Xen Server 5.6 and Ubuntu 11.04 Server KVM Hypervisors using standard benchmark SPECvirt_sc2010v1.01 formulated by Standard Performance Evaluation Corporation (SPEC) under various workloads simulating real life situations.

Chapter 1

INTRODUCTION

Information Technology (IT) has been experiencing exponential growth over the past decade. Initially, IT found use in manufacturing automation and other highly specialized tasks. However, in the past decade, IT has started to enter new regimes like social media and since then it has become a part of everyone‘s daily life.

This increased demand for IT resources has created the enormous challenge of deploying and managing IT infrastructure in a larger scale. While deploying and managing this large scale IT infrastructure is an issue, an even bigger issue is the scaling up of the IT infrastructure. The server is one of the key hardware resources for an IT infrastructure. A typical server infrastructure contains:

1. Server racks on which several servers are mounted.
2. High-speed network switches.
3. Air conditioning system.
4. Uninterruptible power supply (for short-term power outage).
5. Gasoline/Diesel Backup Generator (for long-term power outage).

For businesses whose core competency is not in an IT field, it is a big capital investment to construct and maintain this server infrastructure. For these reasons, businesses have started utilizing server farms hosted by companies that provide these types of IT services. A while back when few businesses were using outsourced servers for IT needs, demand was manageable for the companies that provided server rental services. Due to the recent increased demand, these server rental companies are struggling with the following issues:

1. Large server farms consume a lot of electricity. Due to the increasing price of electricity, server-hosting businesses have started to gain lower profits.
2. Each individual server in a server farm could be underutilized, causing wastage of valuable IT resources.

Cloud computing was designed to address these two issues. The core technology that has made cloud computing possible is hardware virtualization. This piece of the technology is called Hypervisor. Cloud computing utilizes advanced high-performance server systems with large amounts of memory, storage and multiple processors.

Hypervisor creates multiple virtual servers within a single physical server. Each virtual server could have its own operating system (OS) installed in it. Many virtual servers can be operated simultaneously and independently of each other. Hypervisor enables the pooling of the processor and memory resources. Installing a Hypervisor on the host server enables it to run multiple operating systems simultaneously using virtualization technology. By using server virtualization, the number of physical servers could be reduced significantly.

"Virtualization is a technology that combines or divides computing resources to present one or many operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, time-sharing, and many others" [Nanda05].

A virtualization layer provides an infrastructural support using the lower-level resources to create multiple virtual machines that are independent and isolated from each other. Such a virtualization layer is also called Hypervisor. Although traditionally Hypervisor is used to mean a virtualization layer right on top of the hardware and below the operating system, we might use it to represent a generic layer in many cases [Nanda05].

The various virtualization levels of abstraction are instruction set level, hardware abstraction layer (HAL) level, OS level (system call interface), user-level library interface, or in the application level as shown in Figure 1. A virtual machine represents an operating environment for a set of user-level applications, which includes libraries, system call interface/service, system configuration, daemon processes, and file system state. –At any levels of abstraction, the general phenomenon is the same in that it partitions the lower-level resources using some novel techniques to map to multiple higher-level VMs transparently" [Nanda05].

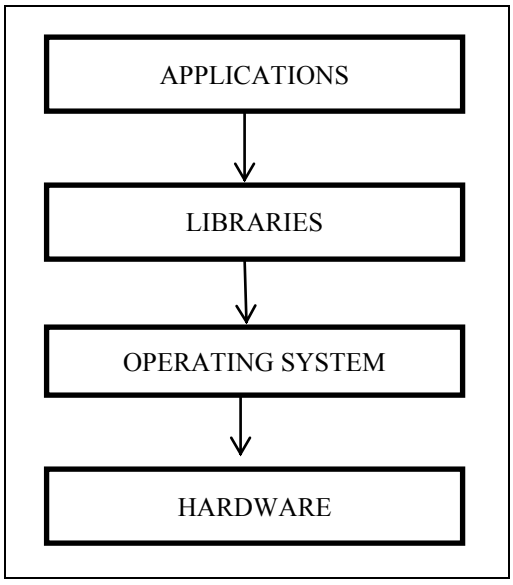

Figure 1: Machine Stack Showing Virtualization Opportunities

―The functionality and abstraction level of a HAL level virtual machine lies between a real machine and an emulator" [Nanda05]. A virtual machine is an environment created by a Hypervisor, which is the virtualization software lying between the bare hardware and the operating system and gives the operating system a virtualized view of all the hardware. A Hypervisor can create multiple virtual machines (VMs) on a single machine. An emulator provides a complete layer between the operating system or applications and the hardware. A Hypervisor manages one or more virtual machines where every virtual machine provides facilities to an operating system or application to run as if it is in a normal environment and directly on the hardware. Virtual machines operating at HAL layer level give the flexibility of using different operating systems or different versions of the same operating system on the same machine by presenting a complete machine interface, which creates a demand for a much greater amount of resources.

Generally, there are two types of Hypervisors:

- Type 1 Hypervisor, which runs directly on the system hardware. This is also known as bare metal approach Hypervisor.
- Type 2 Hypervisor, which runs on host operating system that provides virtualization services such as I/O and memory management. This is also known as a hosted approach Hypervisor.

There are two primary approaches to virtualization:

- Platform virtualization. Ex : Server
- Resources virtualization. Ex : Storage , Network

The three types of virtualization namely server virtualization, storage virtualization and network virtualization are described below:

- Server virtualization is dividing the single physical machine into multiple virtual servers. The main server virtualization categories are full virtualization, para-virtualization and OS-level virtualization. Full virtualization enables Hypervisors to run an unmodified guest operating system and it is not aware that it is being virtualized. Para-virtualization technique involves explicitly modifying the operating system so that it is aware of being virtualized. Operating system level virtualization technique provides efficient architecture with one operating system instance.
- Storage virtualization pools multiple physical storage resources into a single storage resource and it is centrally managed.
- Network virtualization combines the available resources in a network by splitting up the available bandwidth into channels. Channels are independent of each other and each can be assigned or reassigned to a particular server or device in a real time environment.

1.1 VMware ESXi 4.1

VMware ESXi is a type I Hypervisor aimed at server virtualization environments capable of live migration using VM motion and booting VMs from network attached devices. VMware ESXi is a lightweight implementation. VMware ESXi lacks the service console included in VMware ESX. VMware ESXi 4.1 supports full virtualization. Figure 2 shows the architecture of ESXi.

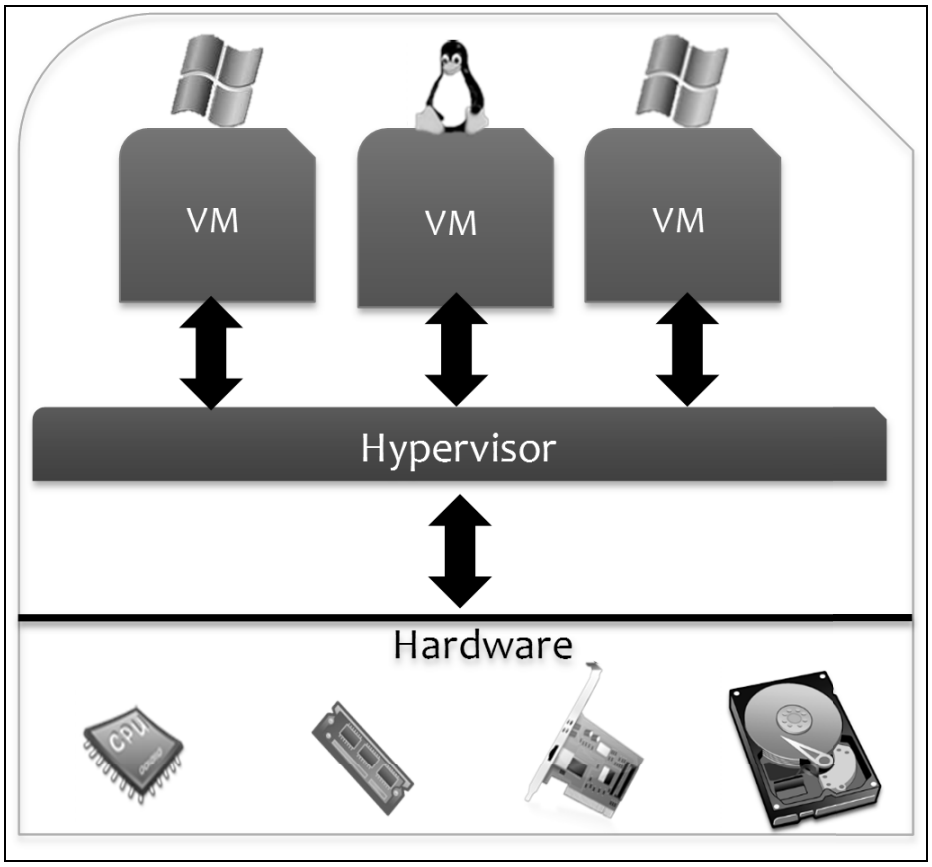

Figure 2: VMware ESXi Architecture

VMware ESXi server product is installed on a bare machine without any operating system. It provides a console interface to create and configure Virtual Machines. Since there is no host operating system, the Hypervisor handles all the I/O instructions, which necessitates the installation of all the hardware drivers and related software. It implements shadow versions of system structures such as page tables and maintains consistency with the virtual tables by trapping every instruction that attempts to update these structures. Hence, an extra level of mapping is in the page table.

The virtual pages are mapped to physical pages throughout the guest operating system‘s page table [Barham03]. The Hypervisor then translates the physical page (often-called frame) to the machine page, which eventually is the correct page in physical memory. This helps the ESXi server better manage the overall memory and improve the overall system performance.

The product typically finds use in server consolidation and web hosting. It uses various other techniques to increase the overall efficiency, and level of isolation to keep the VMs independent from one another, making it a reliable system for commercial deployment.

VMware‘s proprietary ESXi Hypervisor, in the vSphere cloud-computing platform, provides a host of capabilities not currently available with any other Hypervisors. These capabilities include High Availability (the ability to recover virtual machines quickly in the event of a physical server failure), Distributed Resource Scheduling (automated load balancing across a cluster of ESXi servers), Distributed Power Management (automated decommissioning of unneeded servers during non-peak periods), Fault Tolerance (zero-downtime services even in the event of hardware failure), and Site Recovery Manager (the ability to automatically recover virtual environments in a different physical location if an entire datacenter outage occurs) [Hostway11].

1.2 Citrix XENServer 5.6

Citrix XenServer 5.6 is an open-source, complete, managed server virtualization platform built on the powerful Xen Hypervisor. Xen uses para-virtualization. Para-virtualization modifies the guest operating system so that it is aware of being virtualized on a single physical machine with less performance loss. Figure 3 shows the Xen server architecture.

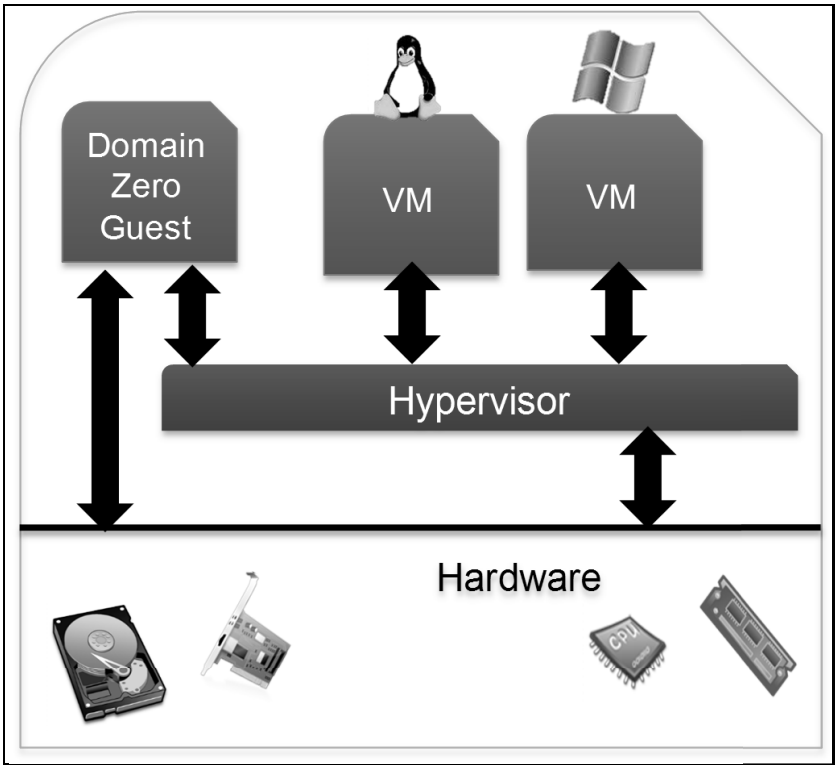

Figure 3: XEN Architecture

XenServer is a complete virtual infrastructure solution that includes a 64-bit Hypervisor with live migration, full management console, and the tools needed to move applications, desktops, and servers from a physical to a virtual environment [Fujitsu10B]. XenServer

creates and manages unlimited servers and virtual machines to run safely and securely from a single management console.

Customers who need additional management, availability, integration, or automation capabilities can upgrade to a premium edition of XenServer to create an enhanced virtual datacenter [Fujitsu10B]. The Advanced, Enterprise, and Platinum Editions of XenServer offer rich management and automation capabilities that provide full datacenter automation, advanced integration and management, and key performance features.

Based on the open-source design of Xen, XenServer is a highly reliable, available, and secure virtualization platform that provides near native application performance [Fujitsu10B]. Xen usually runs in higher privilege level than the kernels of guest operating systems. It is guaranteed by running Xen in ring 0 and migrating guest operating systems to ring 1. When a guest operating system tries to execute a sensitive privilege instruction (e.g., installing a new page table), the processor will stop and trap it into Xen [Che08].

In Xen, guest operating systems are responsible for allocating the hardware page table, but they only have the privilege of direct read, and Xen [Che08] must validate updating the hardware page table. Additionally, guest operating systems can access hardware memory with only non-continuous way because Xen occupies the top 64MB section of every address space to avoid a TLB flush when entering and leaving the Hypervisor [Che08].

As for the page fault, Xen causes an extended stack frame to record the faulting address that should be read from the privileged processor register (CR2). Regarding the exceptions such as system calls, Xen allows each guest Operating system to register a fast exception handler that can be accessed directly by the processor without passing via ring 0. [Che08]

However, this handler is verified before it is installed in the hardware exception table. Xen hosts most unmodified Linux device drivers into an initial domain called Domain0, which plays the role of driver domain. Domain0 is created at boot time and is responsible for the control of creating, pausing, migrating and terminating other domains (guest domains), CPU scheduling parameters and resource allocation policies.

To achieve I/O operation‘s virtualization, Xen proposes a shared memory and asynchronous buffer descriptor ring model based on device channels. In this model, two aspects of factors must be taken into consideration: transferring I/O message and I/O data. Xen provides two communication mechanisms between guest domains or Xen and guest domains: synchronous call using hyper calls (calls to hypervisor which are analogous to system calls in the OS world) to send messages from guest domains to Xen, and asynchronous event using virtual interrupts to send notifications from Xen to guest domains. When the data requested by a guest domain is moved into physical memory, Domain0 will send a virtual interrupt to the corresponding guest domain and exchange

the memory page containing the data with a vacant memory page presented by the guest domain [Che08].

1.3 Ubuntu 11.04 Server KVM

KVM (Kernel-based Virtual Machine) is another open-source Hypervisor using full virtualization apart from VMware. Figure 4 shows the KVM architecture. As a kernel driver added into Linux, KVM enjoys all advantages of the standard Linux kernel and hardware-assisted virtualization.

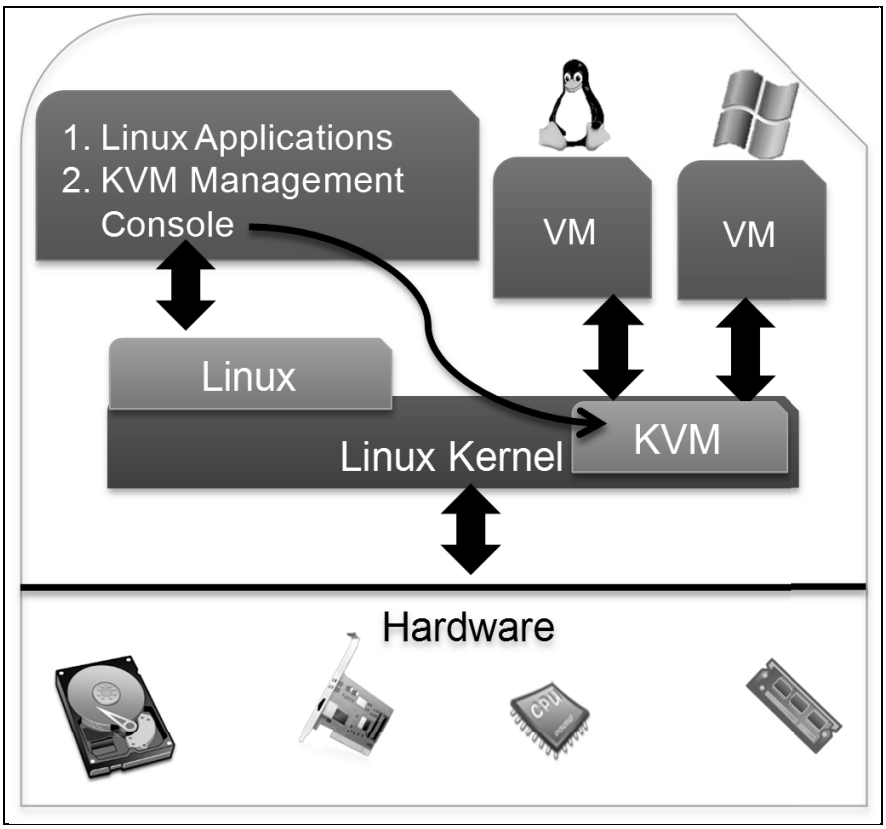

Figure 4: KVM Architecture

KVM introduces virtualization capability by augmenting the traditional kernel and user modes of Linux with a new process mode named guest, which has its own kernel and user modes and answers for code execution of guest operating systems [Che08].

KVM comprises two components: one is the kernel module and another one is user-space. Kernel module (namely kvm.ko) is a device driver that presents the ability to manage virtual hardware and see the virtualization of memory through a character device /dev/kvm. With /dev/kvm, every virtual machine can have its own address space allocated by the Linux scheduler when being instantiated [Che08]. The memory mapped for a virtual machine is actually virtual memory mapped into the corresponding process. Translation of memory address from guest to host is supported by a set of page tables. KVM can easily manage guest Operating systems with kill command and /dev/kvm. User-space takes charge of I/O operation‘s virtualization.

KVM also provides a mechanism for user-space to inject interrupts into guest operating systems. User-space is a lightly modified QEMU, which exposes a platform virtualization solution to an entire PC environment including disks, graphic adapters and network devices [Che08]. Any I/O requests of guest operating systems are intercepted and routed into user mode to be emulated by QEMU [Che08].

Chapter 2

LITERATURE REVIEW

When we survey the literature on "Comparison of Hypervisors," we come across three major ways of comparing the Hypervisors:

1. Based on benchmarks, which compare the impact on overheads on CPU bound, memory bound, I/O bound operations.
2. Based on micro benchmarks, which compare the impact on basic primitive operation, and extending to real life situations by prediction.
3. Based on benchmarks, which compare the performance by generating workloads similar to real life situations.

Based on benchmarks, which compare the impact on overheads on CPU bound, memory bound, I/O bound jobs, the following two papers surveyed come under this category.

In the first paper titled "Performance comparison of Hypervisors" - Performance study by VMware [VMware07A], quantitative comparison of two Hypervisors namely VMware's ESX and XenSource's Xen is done. The following standard benchmark tests were chosen for these experiments:

• SPECcpu2000, The integer component of the benchmark suite available from Standard Performance Evaluation Corporation, represents CPU-intensive applications.

• Passmark, a synthetic suite of benchmarks to isolate various aspects of workstation performance, represents desktop-oriented workloads.

• Netperf, used to simulate the network usage in a datacenter.

• SPECjbb2005, benchmark suite from SPEC used to represent the Java applications used in the datacenters.

• A compile Workload — build SPECcpu2000 INT package, which was also added to capture typical IT development and test usage in datacenters.

The main objective of these benchmarking experiments was to test the performance and scalability of the two virtualization Hypervisors VMware and Xen.

In the second paper titled –Performance comparison of commercial Hypervisor" - A study by XENSOURCE [Xen Source07], a quantitative comparison of the above-mentioned Hypervisors is done. This paper studies the Hypervisor-based virtualization products from VMware and XenSource. Using the VMware ESX Server Hypervisor as an industry benchmark for performance and enterprise readiness, the study presents comparative results from an assessment of XenSource's XenServer virtualization product family using industry standard benchmarks for performance and scalability [Xen Source07].

This second paper presents results for the same performance benchmarks as published by VMware, comparing Xen Source's Xen Enterprise 3.2 commercial products, which are based on Xen 3.0.4 and is bundled with XenSource's Enhancements for virtualized Windows guests, with the commercially licensed ESX 3.0.1. In the study

by Xen Source, Xen Enterprise performs just as well as ESX 3.0.1, and in many cases it had performed better. In a few tests, it performs less well than ESX, these were highlighted as key points for improvement of Xen in later releases [Xen Source07].

The following paper was surveyed in the category where the benchmarking is based on micro benchmarks, which compare the impact on basic primitive operations and extending to real life situations by prediction: This paper is a thesis titled –Virtual Machine Benchmarking," A Diploma thesis by Kim Thomas Moller [Moller07] wherein a new benchmark VMBench is proposed. VMBench uses a three-stage approach to characterize the performance of a virtual machine environment. The stages were built upon each other, increasingly tolerating complexity, non-determinism of the environment.

Stage 1: Hypervisor performance signature

–In the first stage, micro- and nano-benchmarks determine the Hypervisor performance signature, the best-case performance of a virtual machine's primitive operations for a given combination of hardware, Hypervisor, operating system and workload" [Moller07]. Therefore, a single virtual machine exercises well-defined operations, so that the performance of virtualization-specific functional primitives can be measured accurately. To determine the best-case performance, VMBench minimizes the side effects and interprets the results optimistically.

Stage 2:Best-case predictions for realistic applications

The second stage combines the outcome of the virtual machine performance of the first stage using a linear model to predict best-case results for realistic applications [Moller07].

Stage 3: Analysis of VM interference

The third stage examines how the prediction from the second stage varies under non-optimal conditions caused by concurrent virtual machines [Moller07]. VMBench follows a latency-oriented approach rather than data throughput.

The following two papers were surveyed in the category where benchmarks were based on generating workloads similar to real life situations:

The first paper titled ―VMmark - A scalable Benchmark for Virtualized Systems," by Vikram Makhija and Bruce Herndon of VMware [Makhija06], presents a tile-based benchmarking method. This consists of several familiar workloads executing simultaneously in separate virtual machines. Each workload component is based on a single-system benchmark executing at less than full utilization. This collection of different workloads is aggregated into a unit of work referred to as a tile. The performance of each workload is measured and forms an aggregate score for the tile [Makhija06]. The overall benchmark score is calculated by summing up the scores generated when running multi-tiles simultaneously.

—Tile is the unit of work for a benchmark of virtualized consolidation environments and is defined as a collection of virtual machines executing a set of diverse workloads" [Makhija06]. Total number of tiles gives a measure of the systems consolidation capacity of the physical system and the virtualization layer.

The following are the workloads based on relevant datacenter workloads: Mail server, Java server, Standby server, Web server, Database server, File server [Makhija06]. Instead of developing new workloads, existing benchmarks were used wherever possible to avoid redundancy and the implementation effort. It provides a well-understood base upon which to build the benchmark, but the benchmark-required modifications to make it suitable for multiple virtual machines benchmarking since the run rules of many benchmarks were sometimes conflicting with the design goals of VMmark. [Makhija06].

The scoring methodology used in VMmark is described below. Once a VMmark test is completed, each individual workload reports the performance metric as shown in Table 1 [Makhija06]. These metrics are collected at regular intervals during the complete run. —A typical VMmark benchmark test is designed to run for at least three hours with workload metrics reported every 60 seconds. Once all workloads have reached the steady state during a benchmark run, a two-hour measurement interval is taken. This steady-state interval is then divided into three 40-minute sections. For each of the 40-minute sections, the results for the tile are calculated and the median

score of the three sections is selected as the raw score for the tile" [Makhija06]. The median of the sums of the per-tile scores is the raw score for multi-tile runs.

After the benchmark run is completed, the workload metrics are calculated for each tile and are aggregated into a score for that tile. "This aggregation is performed by first normalizing the different performance metrics such as MB/s and database commits/s with respect to a reference system" [Makhija06]. Then, a geometric mean of the normalized scores is calculated as the final score for the tile and the final metric is calculated by summing the resulting per-tile score.

Workload	Metric
Mail server	Actions/minute
Java server	New orders/second
Standby server	None
Web server	Accesses/second
Database server	Commits/second
File server	MB/second

Table 1: Individual VMmark Workload Metrics

In the second paper titled "Benchmark Overview - vServCon" a white paper by FUJITSU [Fujitsu10A], scalability measurements of virtualized environments at Fujitsu Technology Solutions are currently accomplished by means of the internal benchmark

"vServCon" (based on ideas from Intel‘s "vConsolidate"). The abbreviation "vServCon" stands for: "virtualization enables SERVer CONsolidation”.

A representative group of application scenarios is selected in the benchmark. It is started simultaneously as a group of VMs on a virtualization host when making a measurement. Each of these VMs is operated with a suitable load tool at a defined lower load level. All known virtualization benchmarks are thus based on a mixed approach of operating system and applications plus an "idle" or "standby" VM, which represents the inactive phases of a virtualization environment and simultaneously increases the number of VMs to be managed by the Hypervisor [Fujitsu10A]. The term "tile" is the name for such a unit of virtual machines. The load can be increased on a step-by-step basis until the system has reached its performance limit.

VServCon is not a new benchmark but is a framework that consolidates already established benchmarks, as workloads, if necessary in modified form in order to simulate the load of a virtualized consolidated server environment. Three proven benchmarks are used, which cover the application scenarios namely database, application server and web server. Each of the three application scenarios is assigned to one dedicated virtual machine (VM). _Idle VM‘ is added as the fourth VM. These four VMs form a "tile.” In the terminology of "vConsolidate," this would be a "consolidation stack unit" (CSU). Because of the performance capability of the underlying server hardware, it is usually necessary to have started several identical tiles in parallel as part of a measurement in order to achieve a maximum overall performance [Fujitsu 10A].

The result of vServCon is a number, known as a "score," which provides information about the performance of the measured virtualization host. The score reflects the maximum total throughput that can be achieved by running a defined mix that consists of numerous application VMs [Fujitsu10A].

The score is determined from the individual results of the VMs. Each of the three vServCon application scenarios provides a specific benchmark result in the form of application-specific transaction rates for the respective VM. In order to derive a normalized score the individual benchmark results for one tile are observed in relation to the respective results of a reference system. The resulting relative performance values are then suitably weighted and finally added up for all VMs and tiles. The outcome is the vServCon score for this tile [Fujitsu10A]. This procedure is performed for an increasing number of tiles, starting with one tile until there is no further significant increase in this vServCon score. The final vServCon score is then the maximum of the vServCon scores for all tile numbers [Fujitsu10A].

The progression of the vServCon scores for the tile numbers provides useful information about the scaling behavior of the "System under Test." Moreover, vServCon also documents the total CPU load of the host (VMs and all other CPU activities) and, if possible, electrical power consumption [Fujitsu10A].

Chapter 3

RESEARCH METHODOLOGY

Based on the literature review, method 3 is chosen as the benchmarking method. Virtualization is mostly used in server consolidation applications. There is an increase in demand for server virtualization in the implementation of IT infrastructure. For this we will use the standard benchmark developed by SPEC and compare the following Hypervisors:

1) VMware ESXi 4.1
2) Citrix Systems Xen Server 5.6
3) Ubuntu 11.04 Server KVM

The benchmark is designed to achieve maximum performance by running one or more sets of virtual machines simultaneously called ―Tiles.‖ SPECvirt_sc2010 is a standard benchmark based on ―Tile‖ concept.

SPECvirt_sc2010 uses a three-workload benchmark design: a web server, Java Application server, and a mail server workload. The three workloads are derived from SPECweb2005, SPECjAppServer2004, ands SPECmail2008 standard benchmarks. ―All three Workloads drive pre-defined loads against sets of virtualized servers. The SPECvirt_sc2010 harness running on the client side controls the workloads and also implements the SPEC power methodology for power measurement‖ [Spec11]. There are three categories to run SPECvirt_sc2010 [Spec11].

- performance only (SPECvirt_sc2010)
- performance/power for the SUT (SPECvirt_sc2010_PPW)

- performance/power for the Server-only (SPECvirt_sc2010_ServerPPW)

Similar to all other SPEC benchmarks, an extensive set of run rules governs SPECvirt_sc2010 disclosures in order to ensure fairness of results [Spec11]. SPECvirt_sc2010 results are not recommended for sizing or capacity planning and the benchmark does not address multiple host performance or application virtualizations.

3.1 Workload Design

The benchmark suite consists of several SPEC workloads representing applications that are the common targets of virtualization and server consolidation. Each of these standard workloads is designed to match a typical server consolidation scenario's resource requirements for CPU, memory, disk I/O, and network utilization for each workload [Spec11]. The SPEC workloads used are:

- SPECweb2005 - This workload represents a web server, a file server, and an infrastructure server. The SPEC web workload is partitioned into two virtual machines (VMs): a web server and a combined file server and backend server (BeSim). Specifically, the support workload is only used, and the characteristics of the download file are modified.
- SPECjAppserver2004 - This workload represents an application server and backend database server. Specifically, the SPECjAppServer is modified in a way to create a dynamic load, the database scale is increased, and the session lengths are decreased.
- SPECmail2008 - This workload represents a mail server. Specifically, we modified the SPEC mail IMAP with new transactions.

SPEC poll is an additional workload being created. SPEC poll serves two purposes: it sends and acknowledges network pings 1) against the idle server in 100% load phase to measure its responsiveness and 2) to all VMs in the 0% load phase (active idle) during power-enabled runs [Spec11].

When consolidating servers, lightly loaded systems are considered. These systems will still place resource demands upon the virtualization layer even when idle and will affect the performance of other virtual machines [Spec11].

Datacenter workloads are researched thoroughly to determine suitable load parameters. The test methodology is expected to ensure that the results scale up with the capabilities of the system. ―The benchmark does not require that each workload have a maximum number of logical (hardware-wise) processors and is designed to run on a broad range of single host systems" [Spec11].The benchmark requires significant amounts of memory (RAM), storage, and networking in addition to processors on the System under Test. Client systems used for load generation must also be configured well to prevent overload.

3.2 Virtual Machines and Tiles

Figure 5 shows the definition of the tile. Tile consists of six virtual machines and is designed as illustrated below.

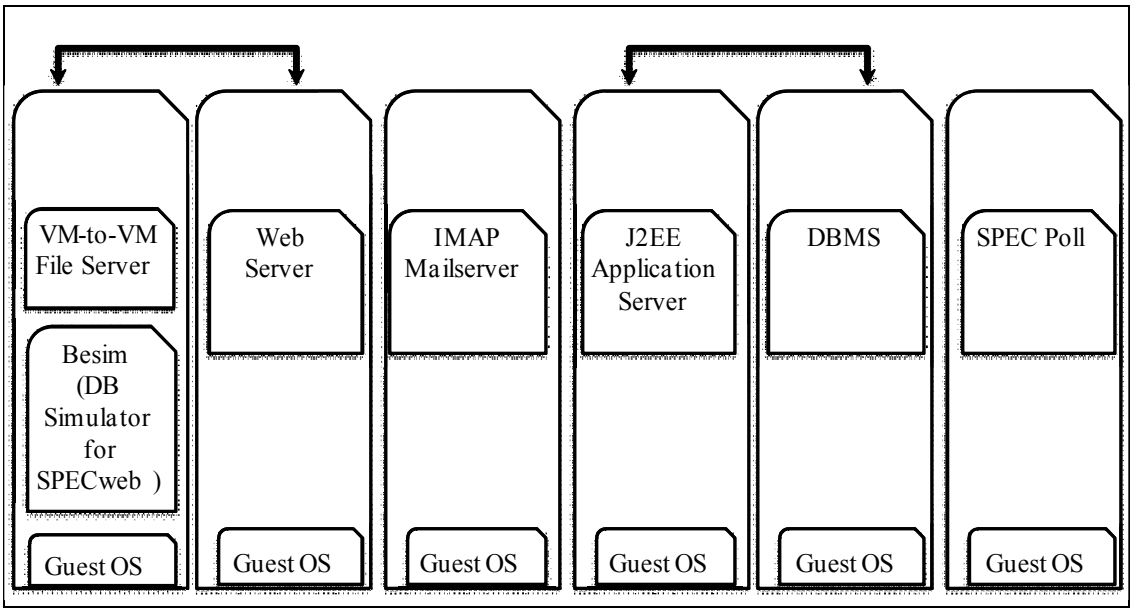

Figure 5: The Definition of a Tile

The web server and infrastructure server share an internal (private) network connection and the application server and database server share an internal (private) network connection to emulate a typical datacenter network use. All virtual machines use an external (public) network to communicate with each other and with the other clients and controller in the test bed. Figure 6 shows the interaction between the tile and the workload.

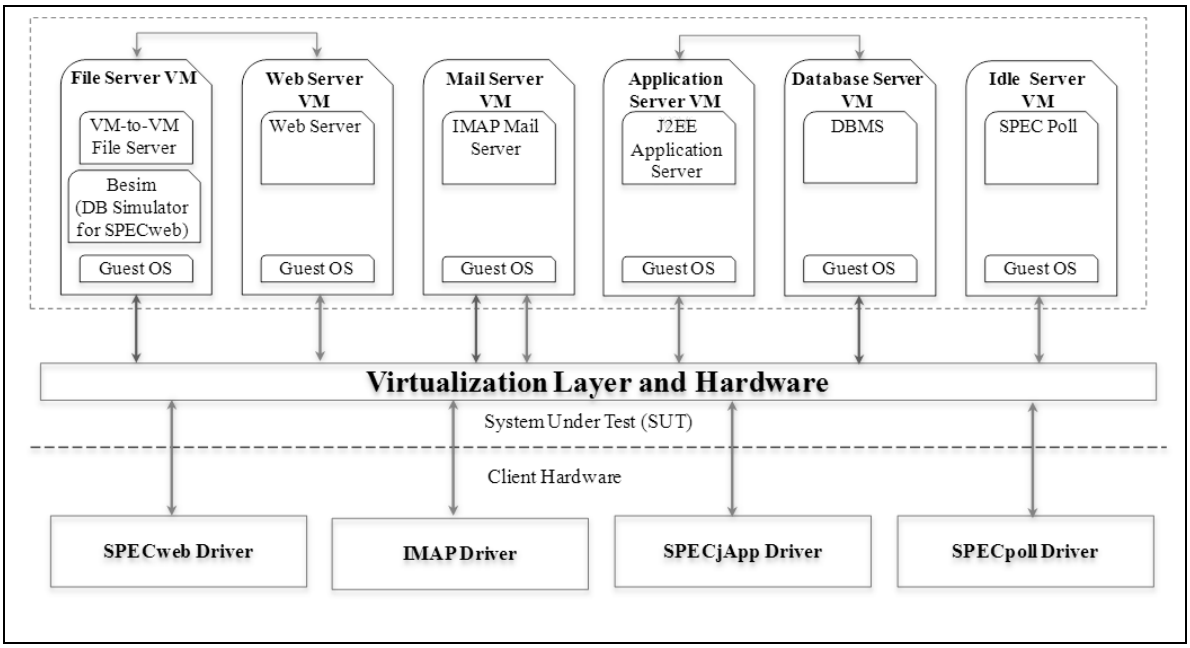

Figure 6: Interaction between the Tile and Harness Workloads

–Scaling the workload on the System under Test consists of running an increasing number of tiles" [Spec11]. Scaling the workload is an important criterion in this benchmark. –Peak performance is the point at which the addition of another tile (or fraction) either fails the Quality of service (QOS) criteria or fails to improve the overall metric –[Spec11]. Figure 7 shows the Multi-tile and client harness configuration.

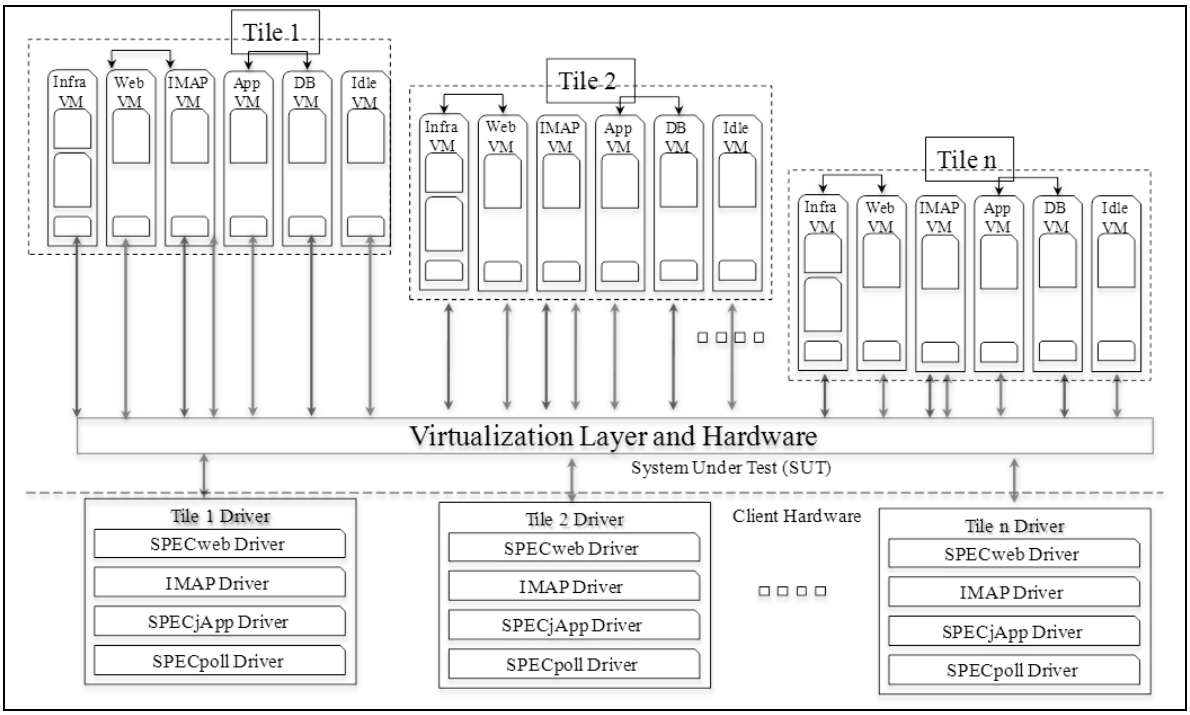

Figure 7: Multi-tile and Harness Configuration

A fractional load tile is used when the System under test does not have sufficient system resources to fully support load of an additional tile because of hardware constraints. A fractional tile consists of an entire tile with all six Virtual Machines but running at a reduced percentage of its full load [Spec11], so that the system performance can be measured when it is completely saturated.

Chapter 4

METRICS AND SUBMETRICS

There are three categories of results supported by SPECvirt_sc2010 benchmark as discussed in the previous section. Each category has different primary metric and the results are compared within that category. This thesis focuses on the first category Performance-Only and its metric is expressed as SPECvirt_sc2010 <Overall_Score> @ <6 * Number_of_Tiles> VMs on the reporting page after the benchmark run is completed [Spec11]. The overall score is based upon the following metrics of the three component workloads: [Spec11]

1. Web server - requests/second at a given number of simultaneous sessions
2. Mail server - the sum of all operations/second at a given number of users
3. Application server - operations/second (JOPS) at a given injection rate, load factor
4. Idle server - msec/network ping (not part of the metric calculation)

—The overall score is calculated by taking each component workload in each tile and normalizing it against its theoretical maximum for the pre-defined load level. The three normalized throughput scores for each tile are averaged arithmetically to create a per-tile sub metric, and the sub metrics of all tiles are added to get the overall performance metric" [Spec11]. The SPECvirt_sc2010 metric reports this overall metric along with the total number of VMs used (6* Number of _Tiles). Each workload receives equal weighting when determining the score. Since the injection load for the three workloads is

fixed (500 web users, 500 mail users, 20IR average jApp load), a theoretical maximum score can be determined.

Control.config:

WORKLOAD_SCORE_TMAX_VALUE is the theoretical maximum throughput rate for each Workload

WORKLOAD_SCORE_TMAX_VALUE [0] = 34.87 (Application server)

WORKLOAD_SCORE_TMAX_VALUE [1] = 54.17 (Web server)

WORKLOAD_SCORE_TMAX_VALUE [2] = 89.93 (Mail server)

WORKLOAD_SCORE_TMAX_VALUE [3] = 0 (Idle server)

Therefore, for example, a score may be calculated as shown below with the results obtained from the SPECvirt_sc2010 benchmarking of application server, web server, mail server: Letting x = Application server, y = Web server and z = Mail server, the per-tile score may be calculated as follows:

(x/34.86 + y/53.72 + z/89.93) / 3 * 100.

The score is calculated by adding all the per-tile scores for multi-tile scores.

Fractional tile is added when the system does not have sufficient resources to support a complete tile. One fractional tile is configured to use one-tenth to nine-tenths of a tile's normal load level. It can be incremented in one-tenths. In this way, the system can be fully saturated under test and accurate metrics can be reported. The sub-metrics must meet the QOS criteria adapted from each SPEC standard workload.

Chapter 5

HARDWARE REQUIREMENTS

5.1 System under Test Hardware and Software Configuration

Hardware configuration used for the system under test (SUT) is very critical for benchmarking of different Hypervisors. Hypervisors, especially bare metal Hypervisors, in general support a limited set of hardware. When benchmarking different Hypervisors on the same hardware, the hardware should be compatible with each Hypervisor. The hardware of the SUT used for this benchmark was able to run ESXi, Xen and KVM Hypervisors without any issues. The hardware configuration of the SUT used in this benchmark is given in Table 2. The Hypervisor was installed on the 500GB hard drive. Each tile‘s virtual machine running the mail server, database server, application server and infra server was installed to a dedicated solid-state drive. Mail server, database server, application server and infra server needs very high data throughput hard drive, which was made possible by using the solid-state drive. The idle server and web server, which does not need high-speed hard drive, was installed to the 500GB 7200RPM SATA2 hard drive.

System Under Test Hardware Configuration (SUT)	
Motherboard	P7P55D-E ASUS Mother Board
Processor	Intel Core i7-875K 2.93GHz
Memory	16384 MB SDRAM
Storage Controllers	Intel P55 Express Chipset Onboard

Storage Drives	1 x 500GB SATA2 7200RPM 3 x 120GB SATA3 SSD
Network Adapters	Intel PWLA8391GT PRO/1000 GT PCI Network Adapter

Table 2: System Under Test Hardware Configuration.

5.2 Client Hardware Configuration

In this benchmark, multi-tiles were run simultaneously. Each tile requires one client computer for running the benchmark. Since provisioning of multiple client hardware could be tedious, as recommended by the SPECvirt benchmark the clients were run in a virtual environment. Xen Hypervisor was used to create and run the virtual clients. The client hardware on which Xen Hypervisor ran is shown in Table 3.

Client Hardware Configuration	
Motherboard	Intel DP67BGB3 Mother Board
Processor	Intel Core i7 2600K 3.4GHz
Memory	16384 MB SDRAM
Storage Controllers	Intel P55 Express Chipset Onboard
Storage Drives	1x 500GB Western Digital
Network Adapters	Intel® 82579V Gigabit Ethernet Controller

Table 3: Client Hardware Configuration

Chapter 6

METHODOLOGY

In this chapter, the systematic procedure involved in the setup of the Hypervisor on the SUT, Hypervisor on the client computer, virtual machines on the SUT and client computer are all described.

6.1 Hypervisor Setup

6.1.1 XenServer Hypervisor Installation on the SUT

Follow the steps below to install the XenServer Hypervisor on the SUT

1. Download the CD image file for the free XenServer by visiting http://www.citrix.com/lang/English/lp/lp_1688615.asp.
2. Burn the image file to a CD.
3. Boot from the CD.
4. Install the Hypervisor to the first hard drive in the SUT.
5. Reboot the SUT.
6. Hypervisor will boot up and will display a configuration window with the IP address for remote access.
7. Open the http://IP and download the setup for XenCenter from another desktop or laptop computer running a windows operating system.
8. Run the XenCenter setup.
9. XenCenter can be used to create VM, modify VM configuration, start/stop/reboot VM.

10. End of installation

6.1.2 VMware ESXi Hypervisor Installation on the SUT

Follow the below steps to install the VMware ESXi Hypervisor

1. Download the CD image file for the free VMware ESXi by visiting https://www.vmware.com/tryvmware/?p=free-esxi&lp=default.
2. Burn the image file to a CD.
3. Boot from the CD.
4. Install the Hypervisor to the first hard drive in the SUT.
5. Reboot the SUT.
6. Hypervisor will boot up and will display a configuration window with the IP address for remote access (IP).
7. Open the http://IP and download the setup for VMware vSphere Client from another desktop or laptop computer running a windows operating system.
8. Run the VMware vSphere Client setup.
9. VMware vSphere Client can be used to create VM, modify VM configuration, start/stop/reboot VM.
10. End of installation

6.1.3 Ubuntu KVM Hypervisor Installation on SUT

Follow the below steps to install the KVM Hypervisor

1. Download the CD image file for the Ubuntu Server from

http://www.ubuntu.com/download/server/download

2. Burn the image file to a CD
3. Boot from the CD
4. Install the Ubuntu server to the first hard drive in the SUT
5. Reboot the SUT
6. Login using the user name created during the setup
7. Execute sudo bash
8. Execute apt-get update
9. Execute apt-get install Ubuntu-desktop
10. Execute reboot
11. Ubuntu desktop environment will boot up
12. Open Synaptic Package Manager and install virt-manager
13. virt-manager can be used to create VM, modify VM configuration, start/stop/reboot VM
14. End of installation

6.1.4 XenServer Hypervisor Installation on Client System

The installation of the XenServer Hypervisor on the Client System is exactly the same steps involved in the installation of the XenServer Hypervisor on the SUT. After the installation of the XenServer Hypervisor, four virtual machines were created with the following configurations:

Two Virtual CPU, 30GB Hard Drive, 3GB memory and 1Gbps Network Card.

Perform the following software configuration on all four of the virtual machines

1. Install Windows 7 32bit Professional
2. Install the SPECvirt_sc2010 v1.01 client software components.
3. Install apache-tomcat-7.0.16 on all of the non-master client virtual machine. Apache-tomcat-7.0.16 is required to run the emulator component of the SPECvirt client software.
4. Install Java SE Runtime Environment 1.6.0_26-b03.

6.2 Java Run Time Environment Installation

Java Runtime Environment (JRE) has to be installed on all the virtual machines on each tile and on the client virtual machines. JRE should be installed before installing any other components of SPECvirt. To install JRE for windows environment visit http://www.java.com/en/download/index.jsp. To install JRE for Linux environment execute the below commands at the command prompt

```
sudo bash
apt-get update
apt-get install openjdk-6-jre-headless
```

6.3 SPEC poll Driver Installation on Virtual Machines Running in SUT

SPEC poll driver needs to be running on all of the virtual machines in the SUT.
Table 4 lists the terminal commands that need to be executed on the VM running on the SUT before each benchmark run.

VM	SPECpoll Driver Startup Command
Appserver	"C:\Program Files (x86)\Java\jre6\bin\java.exe" -jar C:\SPECvirt_sc2010\SPECpoll\pollme.jar -n appserver -p 8001
Dbserver	"C:\Program Files (x86)\Java\jre6\bin\java.exe" -jar C:\SPECvirt_sc2010\SPECpoll\pollme.jar -n dbserver -p 8001
Infraserver	"C:\Program Files (x86)\Java\jre6\bin\java.exe" -jar C:\SPECvirt_sc2010\SPECpoll\pollme.jar -n 1.1.1.2 -p 8001
Webserver	"C:\Program Files (x86)\ Java\ jre6\ bin\ java.exe" -jar C:\SPECvirt_sc2010\SPECpoll\pollme.jar -n webserver -p 8001
Idleserver	C:\WINDOWS\system32\java.exe -jar C:\SPECvirt_sc2010\SPECpoll\pollme.jar -n idleserver -p 8001
Mailserver	java –jar /opt /SPECvirt_sc2010/ SPECpoll/pollme.jar -n mailserver -p 8001

Table 4: SPEC poll Driver Startup Command List.

6.4 Tile Configuration

6.4.1 Infraserver Configuration

6.4.1.1 Virtual Machine

Create a virtual machine by visiting the client manager of the respective Hypervisor. The VM should have a configuration as listed below

- 1 Virtual CPU
- 400MB of memory
- 44GB Hard drive
- 2 network cards

Network Card 1 should be connected to the private network called Net1. Network Card 2 should be connected to the bridged network called Net2. In addition, Install Windows Server 2008 R2 64 Bit and install all the patches and updates. Install Specvirt_sc2010 v1.01.

6.4.1.2 Internet Information Service Installation and Configuration

1. Install Internet Information Service (IIS) by visiting Server Manager→Roles and select IIR role.
2. Change the Default website port to 81 by visiting the IIS configuration manager
3. Change the Path Credential for the default website by visiting the Basic Settings option for the default website
4. Enable Anonymous authentication for the default website

6.4.1.3 BeSim Configuration

1. Copy C:\SPECvirt_sc2010\SPECweb2005\Besim\bin\win32.isapi\Besim.dll to the root web folder c:\inetpub\wwwroot.
2. Add Handler Mapping for the Besim.dll by going to the IIS Manager
3. Set the _Request Requisitions‘ to Execute.
4. Select the besim from the _Handler Mapping‘ list .
5. Select _Edit Feature Permissions‘ and select the read, script and execute permissions.
6. Enable 32bit Applications for the default application pool in the IIS‘s list of application pool.
7. Enable anonymous Authentication for the default web site. Use an admin account for the anonymous authentication.
8. Create a folder named _Share‘ in C:\SPECvirt_sc2010 and share it for network access.
9. Turn off password protected sharing for C:\SPECvirt_sc2010\Share

6.4.2 Web Server Configuration

6.4.2.1 Virtual Machine

Create a virtual machine by visiting the client manager of the respective Hypervisor. The VM should have configuration as listed below:

- 1 Virtual CPU
- 800MB of memory

- 40 GB Hard drive
- 2 network cards

Network Card 1 should be connected to the private network called Net1. Network Card 2 should be connected to the bridged network called Net2. In addition, Install Windows Server 2008 R2 64 Bit and install all the patches and updates. Install SPECvirt_sc2010 v1.01.

6.4.2.2 Internet Information Service Installation and Configuration

1. Install IIS by visiting Server Manager→Roles and select IIR role.
2. Turn off the _Known Extensions' feature by adding *, application/octet-stream to the global MIME in the IIS configuration manager.

6.4.2.3 PHP Installation as required by SPECvirt

1. Download php-5.3.6-nts-Win32-VC9-x86.msi by visiting http://www.php.net
2. Run php-5.3.6-nts-Win32-VC9-x86.msi and select _IIS Fast CGI Mode' for the installation.
3. Use a text editor to create test.php file in the root folder of the IIS Webserver website folder with the following content <? php phpinfo(); ?>
4. Open the link http://localhost/test.php to check if php is functional
5. Copy the PHP scripts from the specvirt folder by issuing the below command xcopy C:\SPECvirt_sc2010\SPECweb2005\Scripts\PHP*.* C:\inetpub\wwwroot\ /E

6. Create a symbolic link to the infraserver‘s shared folder by issuing the below command from command prompt

   mklink /D C:\inetpub\wwwroot\support\downloads \\1.1.1.2\Share

6.4.2.4 Infraserver Shared Folder Workload Files Generation

1. Open a command prompt windows in Windows Server
2. Open support_image_props.rc and Support_downloads_props.rc files located in C:\SPECvirt_sc2010\SPECweb2005\wafgen using windows notepad and then change the properties as below:

   TILEINDEX=0

   DOCROOT=c:/inetpub/wwwroot
3. Run the Wafgen.bat from command prompt with the syntax shown below. This will take few seconds to finish.

   C:\SPECvirt_sc2010\SPECweb2005\wafgen\Wafgen.bat

   C:\SPECvirt_sc2010\SPECweb2005\wafgen\windows\support_image_props.rc
4. Run the Wafgen.bat from command prompt with the syntax shown below. This will take about 1hr to finish.

   C:\SPECvirt_sc2010\SPECweb2005\wafgen\Wafgen.bat

   C:\SPECvirt_sc2010\SPECweb2005\wafgen\windows\

   Support_downloads_props.rc

6.4.2.5 Testing Besim Running on Infraserver

1. In windows command prompt run cd C:\SPECvirt_sc2010\SPECweb2005\Besim

2. Test Besim: perl test_besim_support.pl http://infraserver:81/besim.dll
3. Make sure that the output is similar to the ‗Besim Output' shown in appendix

6.4.3 DB Server

6.4.3.1 Virtual Machine

Create a virtual machine by visiting the client manager of the respective Hypervisor. The VM should have configuration as listed below

- 1 Virtual CPU
- 1024MB of memory
- 22 GB Hard drive
- 1 network card

Network Card 1 should be connected to the bridged network called Net2.Also, Install Windows Server 2008 R2 64 Bit and install all the patches and updates. Install SPECvirt_sc2010 v1.01.

6.4.3.2 MySQL Database Server Software Setup

Install MySQL by following these steps:

1. Download MySQL setup file from http://www.mysql.com/.
2. Install MySQL by running the setup file.
3. Reboot the VM.
4. End of installation.

6.4.3.3 MySQL WorkBench Setup

MySQL server can be remotely monitored and administered by using MySQL Work Bench available from http://www.mysql.com/. This software may be installed on the computer that has the Hypervisor management software installed.

6.4.4 AppServer

6.4.4.1 Virtual Machine

Create a virtual machine by visiting the client manager of the respective Hypervisor. The VM should have configuration as listed below:

- 1 Virtual CPU
- 1024MB of memory
- 13 GB Hard drive
- 1 network card

Network Card 1 should be connected to the bridged network called Net2.Also, Install Windows Server 2008 R2 64 Bit and install all the patches and updates. Install SPECvirt_sc2010 v1.01.

6.4.4.2 Application Server Software

1. Download Glass Fish Application Server from http://glassfish.java.net/public/downloadsindex.html#top
2. Install Glass Fish Application Server

3. SPECjAppServer.ear is the Java enterprise archive file that is served by the Glass Fish server. Download this file from disclosure documents available on the SPECvirt website.

4. Open the GlassFish admin console by opening http://localhost:4848. Deploy the SPECjAppServer.ear by going to Applications→Enterprise Applications→Deploy

6.4.5 Mail Server

6.4.5.1 Virtual Machine

Create a virtual machine by visiting the client manager of the respective Hypervisor. The VM should have configuration as listed below:

- 1 Virtual CPU
- 512MB of memory
- 22 GB Hard drive
- 1 network card

6.4.5.2 IMAP Mail Service Configuration

Follow the steps given below to setup the IMAP Mail Server

1. Install Ubuntu server in a virtual machine.

The server needs to be configured with a static IP address for the primary network card. Refer to https://help.ubuntu.com/10.04/serverguide/C/network-configuration.html get more information on configuring the static IP address. The list of static address that needs

to be used for the mail server is given in Table 7.

2. If required, the host name can be changed by using the terminal command

   ```
   vi /etc/hostname
   ```

3. To install the IMAP email server DOVECOT Issue the following commands on a terminal

   ```
   sudo bash
   apt-get update
   apt-get install samba smbfs
   apt-get install dovecot-imapd
   ```

4. Make the content of /etc/dovecot/dovcot.conf as shown below:

   ```
   maildir_very_dirty_syncs = yes
   #mail_fsync = never
   #login_processes_count = 50
   #max_mail_processes = 600
   protocols = imap
   log_path=/var/log/dovecot.log
   info_log_path = /var/log/dovecot-info.log
   ssl = no
   disable_plaintext_auth = no
   mail_location = maildir:~/Maildir
   auth_verbose = yes
   auth default {
   mechanisms = plain
   ```

```
passdb passwd-file {
args = /etc/dovecot/passwd
}
userdb static {
args = uid=avr gid=avr home=/home/avr/%u
}
}
```

5. Make the content of /etc/dovecot/passwd as shown below:

```
a1:{PLAIN}test
a2:{PLAIN}test
a3:{PLAIN}test
a4:{PLAIN}test
a5:{PLAIN}test
.
.
.
.
a495:{PLAIN}test
a496:{PLAIN}test
a497:{PLAIN}test
a498:{PLAIN}test
a499:{PLAIN}test
a500:{PLAIN}test
```

6. Install Java by issuing `apt-get install openjdk-6-jre-headless`
7. Restart Ubuntu

6.5 Host.txt Address configuration of the Clients

The contents of the hosts.txt file in the operating system of each of the client virtual machine needs to be configured properly in order for the client virtual machines to connect to the virtual machines running in the SUT. Table 5 shows the content of the hosts.txt file for the client virtual machines.

Virtual Machine	**Hosts.txt file Content**
Client – Master	192.168.0.100 appserver1-ext 192.168.0.110 dbserver1-int 192.168.0.101 appserver2-ext 192.168.0.111 dbserver2-int 192.168.0.102 appserver3-ext 192.168.0.112 dbserver3-int 192.168.0.130 infraserver1-ext 192.168.0.140 webserver1-ext 192.168.0.131 infraserver2-ext 192.168.0.141 webserver2-ext 192.168.0.132 infraserver3-ext 192.168.0.142 webserver3-ext
Client – VM1	192.168.0.100 appserver appserver1-ext

	192.168.0.110 dbserver dbserver1-int 192.168.0.130 infraserver infraserver1-ext 192.168.0.140 webserver webserver1-ext 192.168.0.150 mailserver mailserver1-ext 192.168.0.120 idleserver idleserver1-ext
Client – VM2	192.168.0.101 appserver appserver1-ext 192.168.0.111 dbserver dbserver1-int 192.168.0.131 infraserver infraserver1-ext 192.168.0.141 webserver webserver1-ext 192.168.0.151 mailserver mailserver1-ext 192.168.0.121 idleserver idleserver1-ext
Client – VM3	192.168.0.102 appserver appserver1-ext 192.168.0.112 dbserver dbserver1-int 192.168.0.132 infraserver infraserver1-ext 192.168.0.142 webserver webserver1-ext 192.168.0.152 mailserver mailserver1-ext 192.168.0.122 idleserver idleserver1-ext

Table 5: Hosts.txt File Configuration of the Client Virtual Machines.

6.6 Host.txt Configuration of Virtual Machines Running on the SUT

Table 6 shows the contents of the hosts.txt file located in the virtual machines running on the SUT.

Tile	VM	Hosts.txt
1	Appserver	192.168.0.100 appserver appserver1-ext 192.168.0.110 dbserver dbserver1-int 192.168.0.196 suganya
2	Appserver	192.168.0.101 appserver appserver2-ext 192.168.0.111 dbserver dbserver2-int 192.168.0.197 suganya
3	Appserver	192.168.0.102 appserver appserver3-ext 192.168.0.112 dbserver dbserver3-int 192.168.0.198 suganya
1	Dbserver	192.168.0.110 dbserver
2	Dbserver	192.168.0.111 dbserver
3	Dbserver	192.168.0.112 dbserver
1	Webserver	192.168.0.140 webserver webserver1-ext 1.1.1.2 infraserver infraserver1-ext 192.168.0.196 suganya
2	Webserver	192.168.0.141 webserver webserver2-ext 1.1.1.2 infraserver infraserver2-ext 192.168.0.197 suganya

3	Webserver	192.168.0.142 webserver webserver3-ext 1.1.1.2 infraserver infraserver3-ext 192.168.0.198 suganya
1	Idleserver	192.168.0.120 idleserver
2	Idleserver	192.168.0.121 idleserver
3	Idleserver	192.168.0.122 idleserver
1	Mailserver	192.168.0.150 mailserver mailserver1
2	Mailserver	192.168.0.151 mailserver mailserver1
3	Mailserver	192.168.0.152 mailserver mailserver1

Table 6: Host.txt File Configuration for the Virtual Machines in Each Tile

6.7 Network IP address configuration of the virtual machines running on the SUT

Table 7 shows the list of static IP addresses used for each of the virtual machine that was running on the SUT.

Tile ID	**Virtual Machine**	**Network Card Index**	**IP Address**
1	Appserver	1	192.168.0.100
2	Appserver	1	192.168.0.101
3	Appserver	1	192.168.0.102
1	Dbserver	1	192.168.0.110

2	Dbserver	1	192.168.0.111
3	Dbserver	1	192.168.0.112
1	Infraserver	1	1.1.1.2
2	Infraserver	1	1.1.1.2
3	Infraserver	1	1.1.1.2
1	Infraserver	2	192.168.0.130
2	Infraserver	2	192.168.0.131
3	Infraserver	2	192.168.0.132
1	Webserver	1	192.168.0.140
2	Webserver	1	192.168.0.141
3	Webserver	1	192.168.0.142
1	Webserver	2	1.1.1.1
2	Webserver	2	1.1.1.1
3	Webserver	2	1.1.1.1
1	Mailserver	1	192.168.0.150
2	Mailserver	1	192.168.0.151
3	Mailserver	1	192.168.0.152
1	Idleserver	1	192.168.0.120
1	Idleserver	1	192.168.0.121
1	Idleserver	1	192.168.0.122

Table 7: Network IP Address Configuration

6.8 Running the Benchmark

In order to run the benchmark a series of steps needs to be carried out in a proper sequence. The steps involved in starting the benchmark run are listed below:

1. Start all the virtual machines belonging to each of the tile.
2. Start the services (web, database, application) etc. within each of the virtual machine running on the SUT.
3. Make sure that the SPEC poll drive is running on each of the virtual machine as configured.
4. Synchronize the clock on all the virtual machines running on the client computer and SUT.
5. Control.config located in the SPECvirt client installation file needs to be changed to specify the number of tiles and the value of the partial workload. This could be accomplished by changing the below two properties in the control.config file

   NUM_TILES = 2

   LOAD_SCALE_FACTORS [2] = "0.5"
6. Issue the following command on each of the client virtual machines

   set CATALINA_BASE=C:\apache-tomcat-7.0.16

   start %CATALINA_HOME%\bin\catalina.bat start

   cd C:\SPECvirt_sc2010\SPECvirt

   start java -jar clientmgr.jar -p 1098 -log

   start java -jar clientmgr.jar -p 1096 -log

   start java -jar clientmgr.jar -p 1094 -log

start java -jar clientmgr.jar -p 1092 -log

start java -jar clientmgr.jar -p 1088 -log

7. Issue the following command on the master client virtual machine

   cd C:\SPECvirt_sc2010\SPECvirt

   start java -jar specvirt.jar –l

Chapter 7

RESULTS AND DISCUSSION

7.1 Quantitative Comparison

7.1.1 Point of Saturation

A server depending on its hardware resources can run multiple tiles, single tile, or partial tile workload. The maximum number of tiles that a Hypervisor can run successfully on a given hardware is a very strong key performance indicator (KPI) and is called Points of Saturation (POS) in this thesis. POS can be used to compare the performance among different Hypervisors, provided the SUT configuration is the same for all the benchmark runs.

Figure 8 shows the POS for the Hypervisors under test. The POS indicated in Figure 8 is the maximum number of tiles the Hypervisor can run without failing the SPECvirt benchmark test run. ESXi was the best performing with the ability to run 2.8 tiles workload. XenServer was below ESXi with a maximum workload of 2.6. KVM is the least performing of all Hypervisors with a score of 0.9 tile workload**.** KVM failed the benchmark run for one tile running at 100% load. SPECvirt did not output any numerical scores since KVM could not finish the benchmark for one tile run at 100% workload. Due to this, KVM was run with one tile at 90% workload to get an idea about the quantitative performance scores. In the following sections of this thesis for KVM only one data point that is available from the benchmark run is reported and should not be interpreted as an overlapped curve in the plots.

The Overall Performance Score (OPS) of the Hypervisors at POS is shown in Figure 9 . ESXi has the best OPS of 271, closely followed by Xen at 252. KVM has the least OPS of 95. Quality of Service (QOS) at POS for each of the Hypervisor under test is shown in Figure 10. QOS at POS is almost the same for ESXi and Xen. QOS at POS for KVM is about 1% lower than ESXi and Xen.

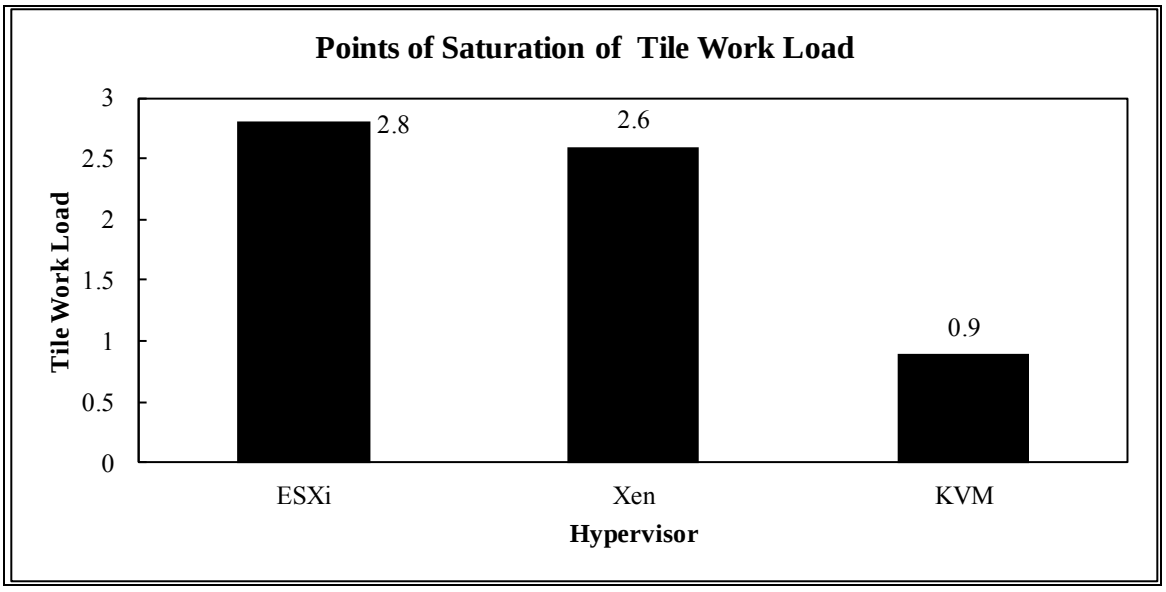

Figure 8: Points of Saturation of Tile Workload.

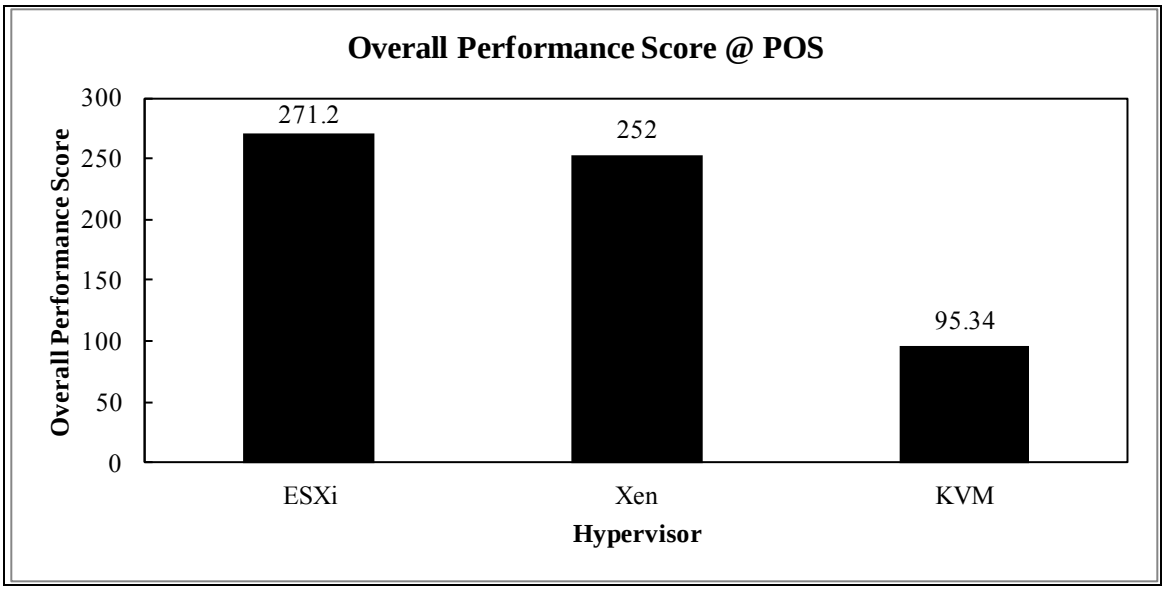

Figure 9: Overall Performance Score at Point of Saturation.

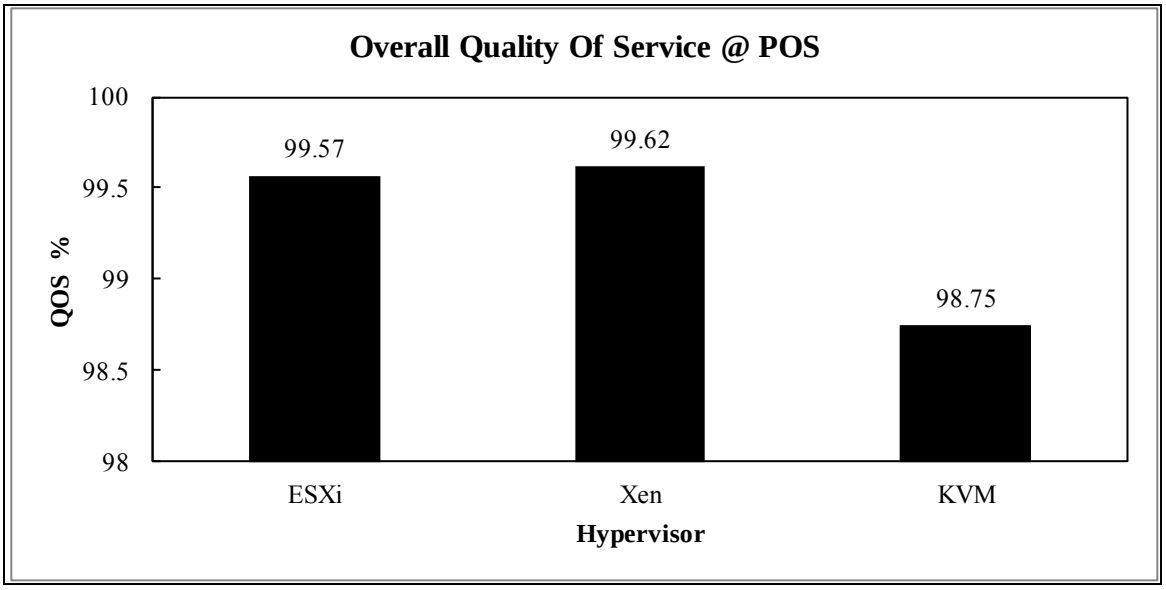

Figure 10: Overall Quality of service at Point of Saturation.

7.1.2 Overall Performance Score and Quality of Service at different Workloads

Figure 11 shows the overall performance score for different workloads for each Hypervisor. At a workload of 1.0, all three Hypervisors had very similar performance score. For workloads of 2.0 and 2.5, both ESXi and Xen had the same performance score. For workload above 2.5, ESXi outperformed Xen. Figure 10 shows the workload versus the quality of service for each of the Hypervisor under test. The quality of service was least for KVM at a workload of 1.0. QOS for both ESXi and Xen was very good for workload range of 1.0 to 2.5. For workload above 2.5, ESXi outperformed Xen. The data used in Figure 11 and Figure 12 are tabulated in Table 8 and Table 9 respectively.

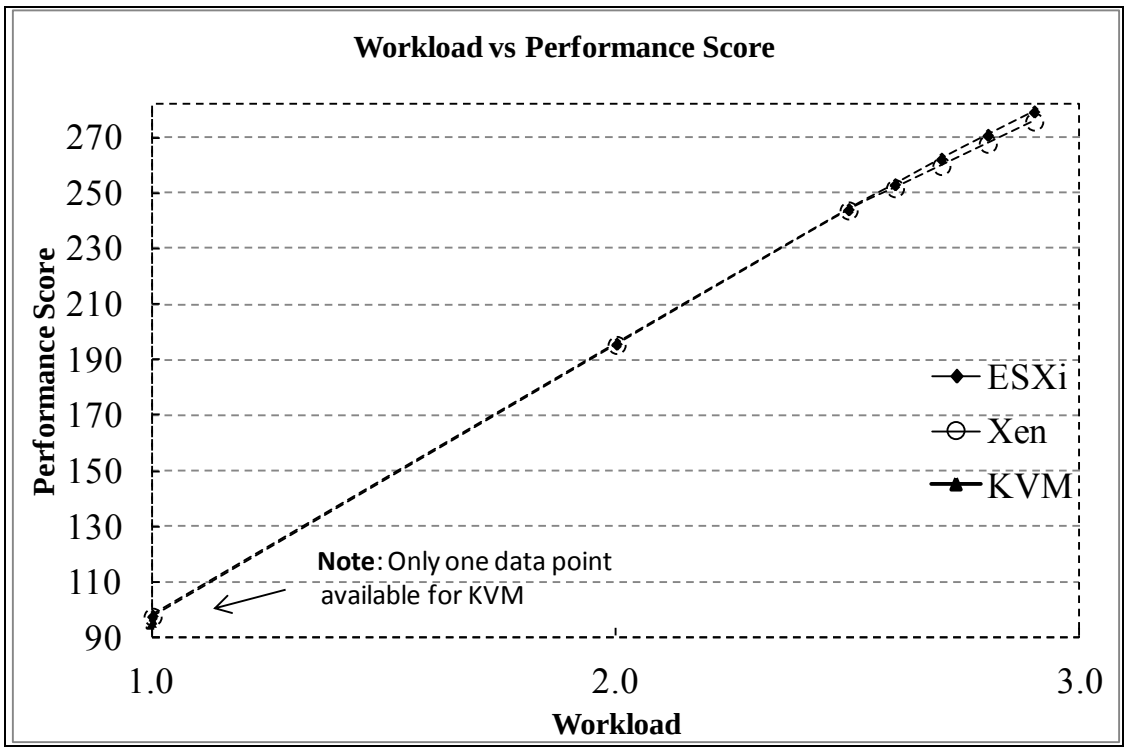

Figure 11: Workload vs. Performance Score.

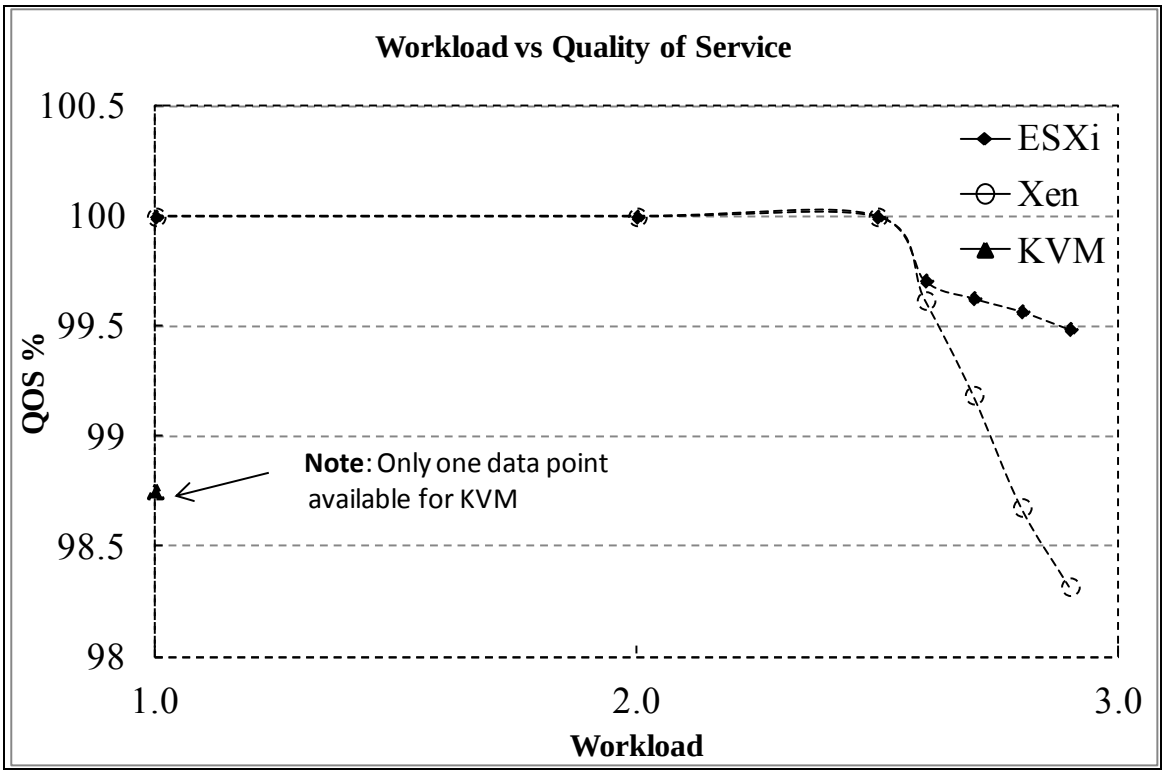

Figure 12: Workload vs. Quality of Service.

	SPEC Performance Score and Compliance					
	ESXi		Xen		KVM	
Workload	**Score**	**SPEC Compliant**	**Score**	**SPEC Compliant**	**Score**	**SPEC Compliant**
1.0	98.07	Yes	97.77	Yes	95.34	No
2.0	196	Yes	195.6	Yes	NA	NA
2.5	244.4	Yes	244	Yes	NA	NA
2.6	253.4	Yes	252	Yes	NA	NA
2.7	262.6	Yes	260	No	NA	NA
2.8	271.2	Yes	268	No	NA	NA
2.9	279.6	No	276	No	NA	NA

Table 8: SPEC Performance score and Compliance of Hypervisors at Different Workloads.

	SPEC QOS % and Compliance					
	ESXi		**Xen**		**KVM**	
Workload	**QOS %**	**SPEC Compliant**	**QOS%**	**SPEC Compliant**	**QOS%**	**SPEC Compliant**
1.0	100	Yes	100	Yes	98.75	No
2.0	100	Yes	100	Yes	NA	NA
2.5	100	Yes	100	Yes	NA	NA
2.6	99.71	Yes	99.62	Yes	NA	NA
2.7	99.63	Yes	99.19	No	NA	NA
2.8	99.57	Yes	98.68	No	NA	NA
2.9	99.49	No	98.32	No	NA	NA

Table 9: SPEC QOS Percentage and Compliance of Hypervisors at Different Workloads

7.1.3 Tile Performance and QOS at Various Workloads

Figure 13 through Figure 26 show the individual tile performance and QOS for workloads ranging from 1.0 to 2.9. Individual tile performance is the best for ESXi at all workloads. Xen‘s tile performance was slightly lower than that of ESXi. KVM‘s performance was well below that of ESXi and Xen. ESXi has the best QOS for the whole range of workload. Xen‘s QOS was very good and comparable to that of ESXi for workloads below 2.7.

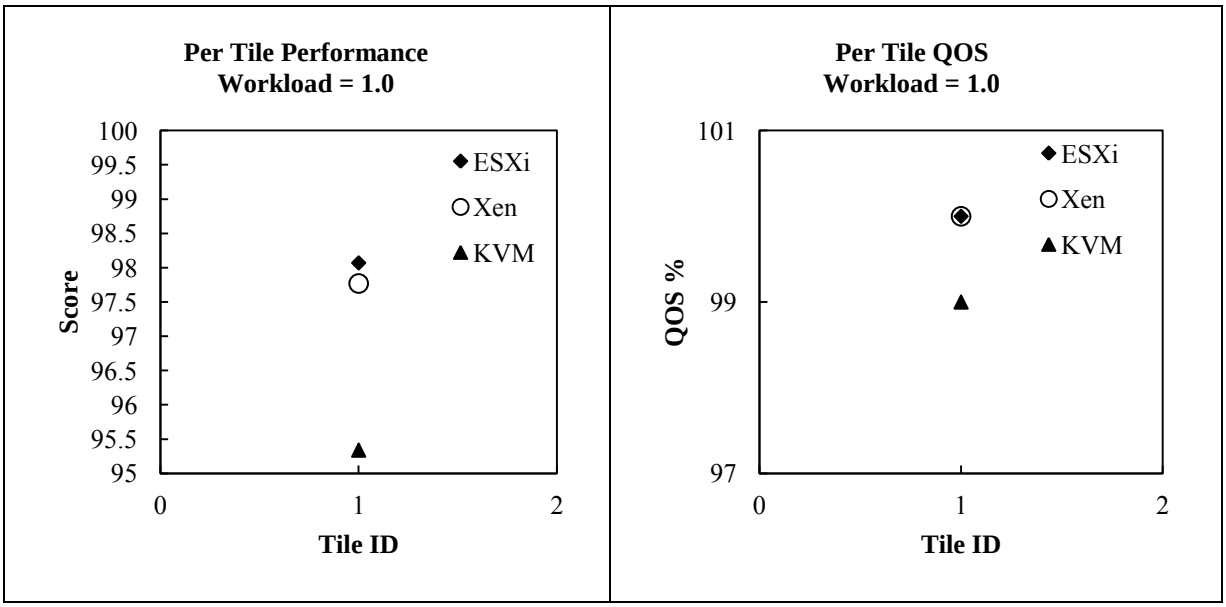

Figure 13: Tile Performance at 1.0 Workload.

Figure 14: Tile Quality of Service at 1.0 Workload.

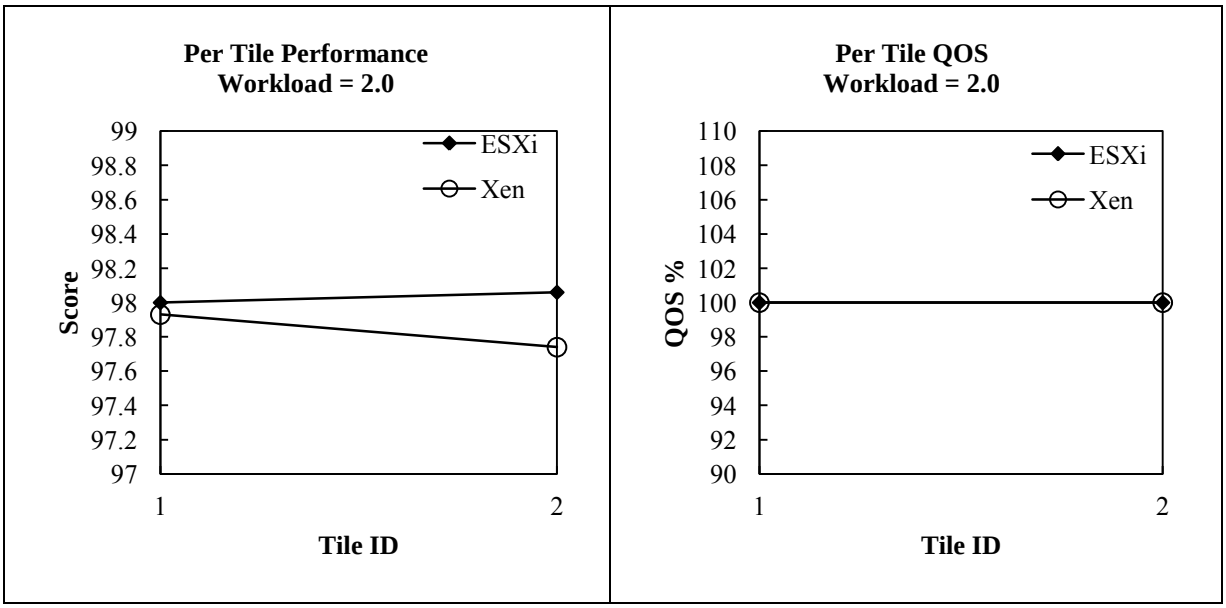

Figure 15: Tile Performance at 2.0 Workload.

Figure 16: Tile Quality of Service at 2.0 Workload.

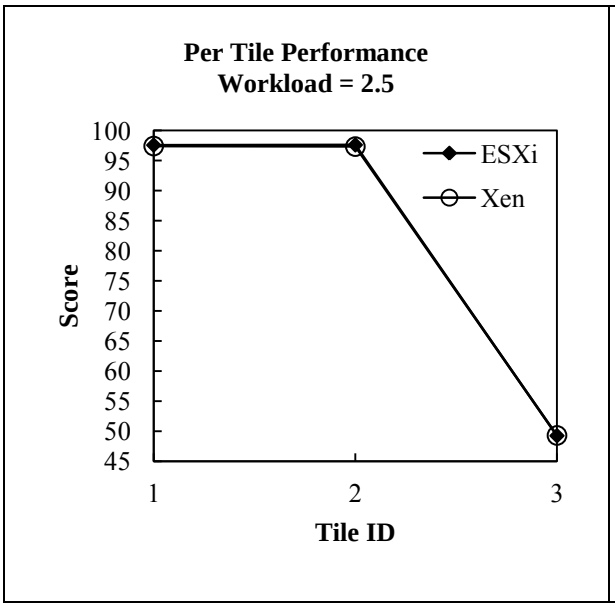

Figure 17: Tile Performance at 2.5 Workload.

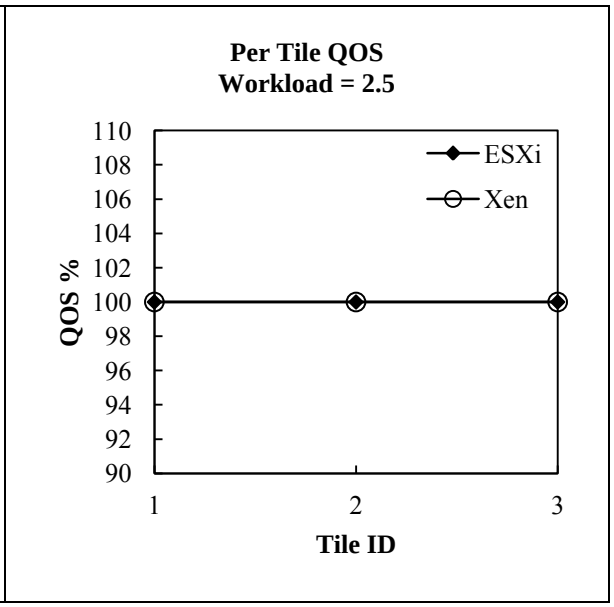

Figure 18: Tile Quality of Service at 2.5 Workload.

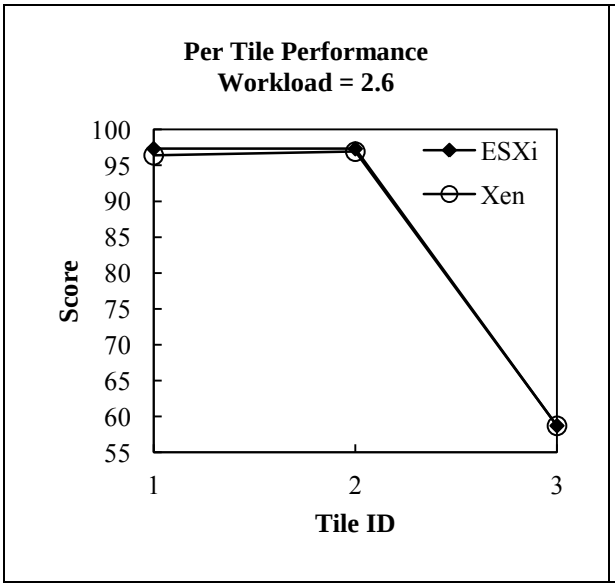

Figure 19: Tile Performance at 2.6 Workload.

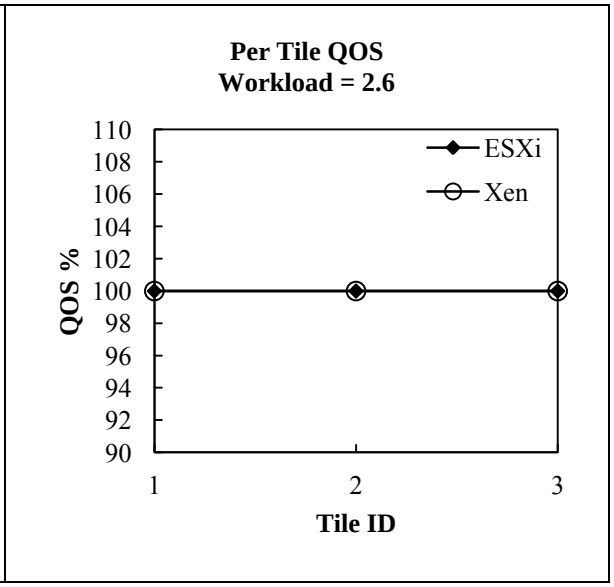

Figure 20: Tile Quality of Service at 2.6 Workload.

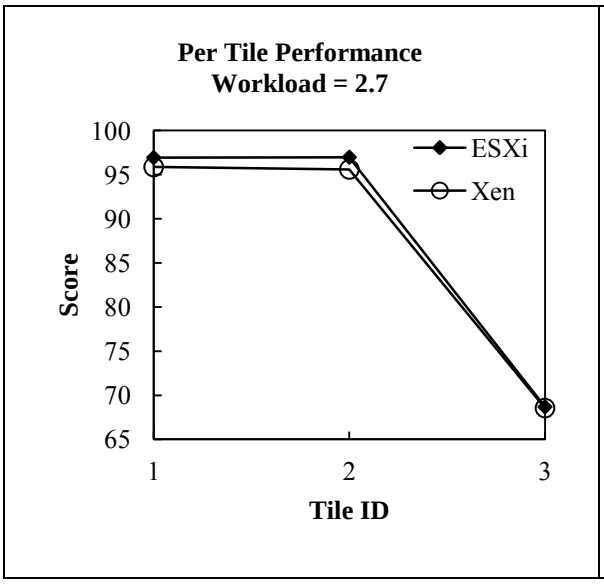

Figure 21: Tile Performance at 2.7 Workload.

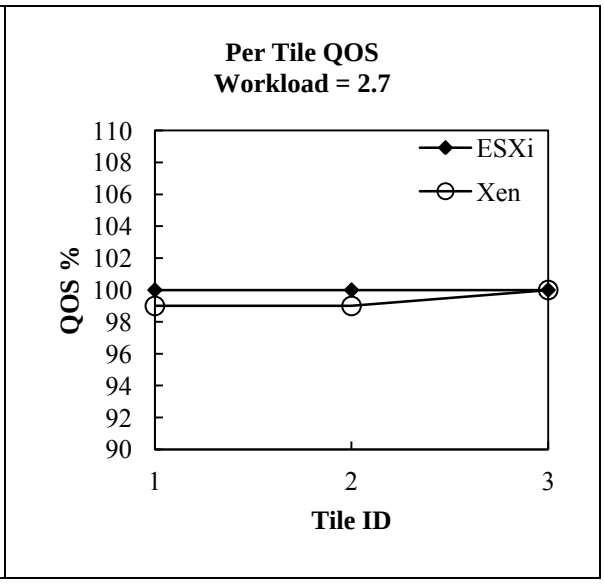

Figure 22: Tile Quality of Service at 2.7 Workload.

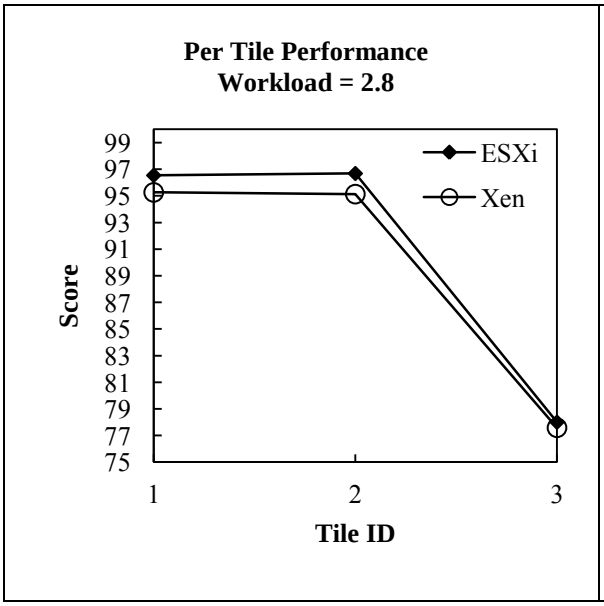

Figure 23: Tile Performance at 2.8 Workload.

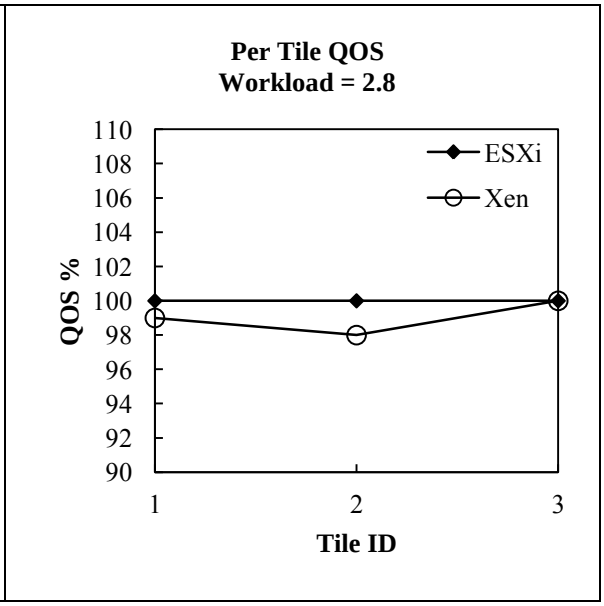

Figure 24: Tile Quality of Service at 2.8 Workload.

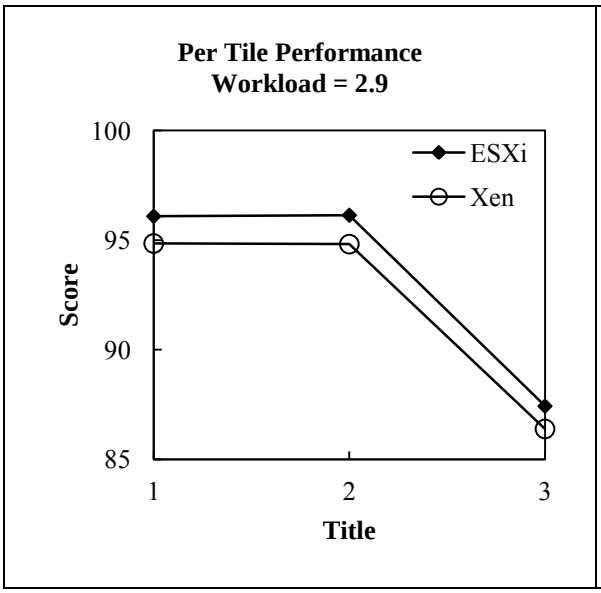

Figure 25: Tile Performance at 2.9 Workload.

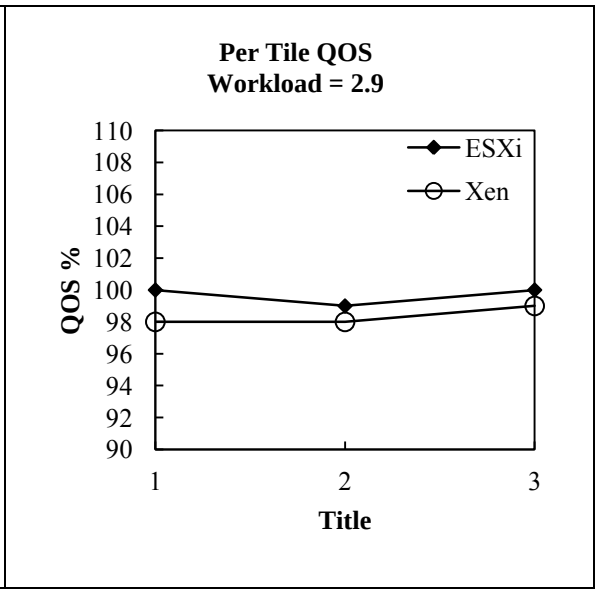

Figure 26: Tile Quality of Service at 2.9 Workload.

7.1.4 Web server performance and QOS by Tile ID

Figure 27 through Figure 39 show the individual web server performance and QOS for workloads ranging from 1.0 to 2.9. Individual web server performance is the best for ESXi at all workloads. Xen‘s web server performance was slightly lower than that of ESXi. KVM‘s performance was well below that of ESXi and Xen. ESXi has the best QOS for the whole range of workload. Xen‘s QOS was very good and comparable to that of ESXi for workloads below 2.7.

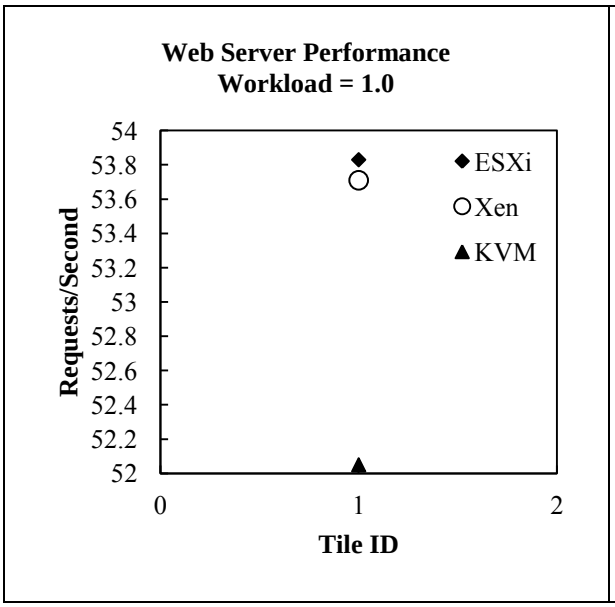

Figure 27: Web Server Performance at 1.0 Workload.

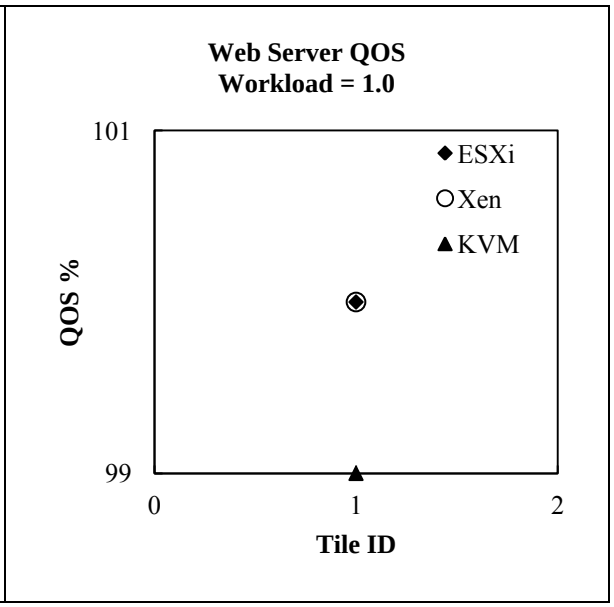

Figure 28: Web Server QOS at 1.0 Workload.

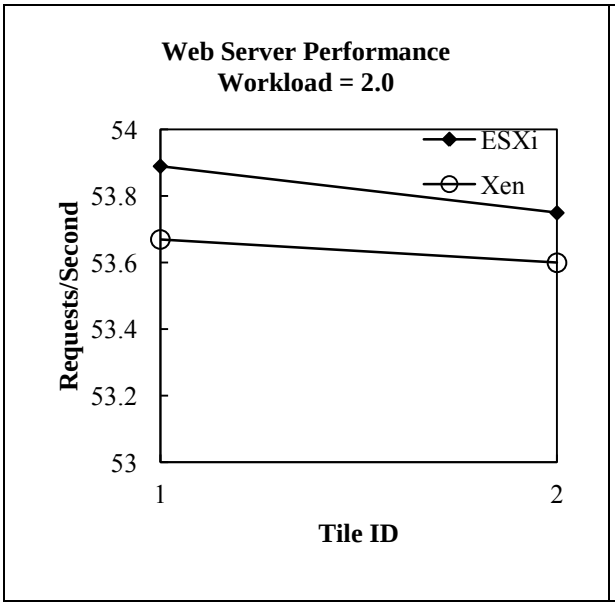

Figure 29: Web Server Performance at 2.0 Workload.

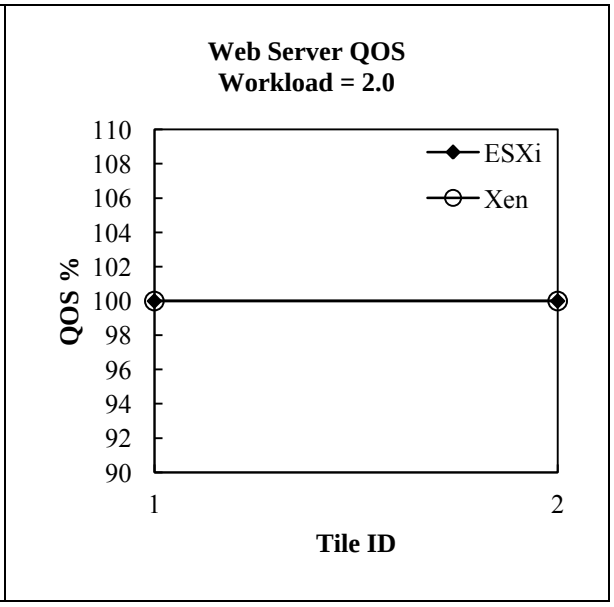

Figure 30: Web Server QOS at 2.0 Workload.

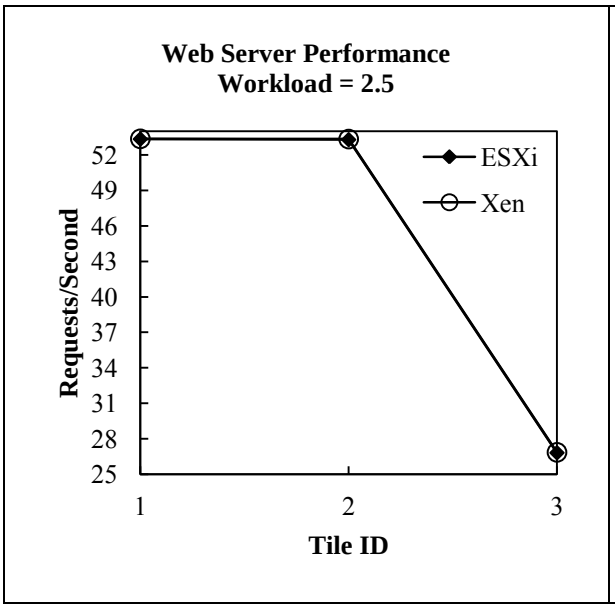

Figure 31: Web Server Performance at 2.5 Workload.

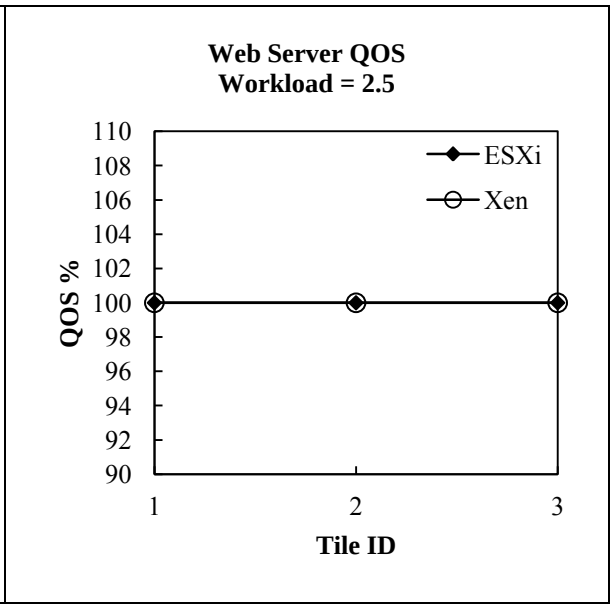

Figure 32: Web Server QOS at 2.5 Workload.

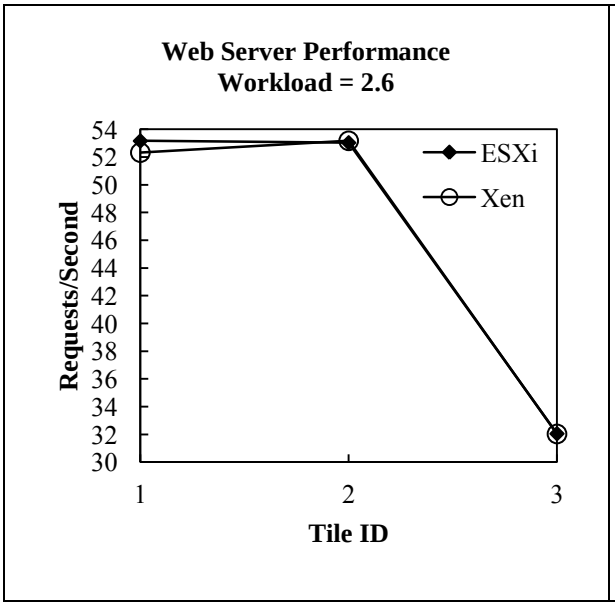

Figure 33: Web Server Performance at 2.6 Workload.

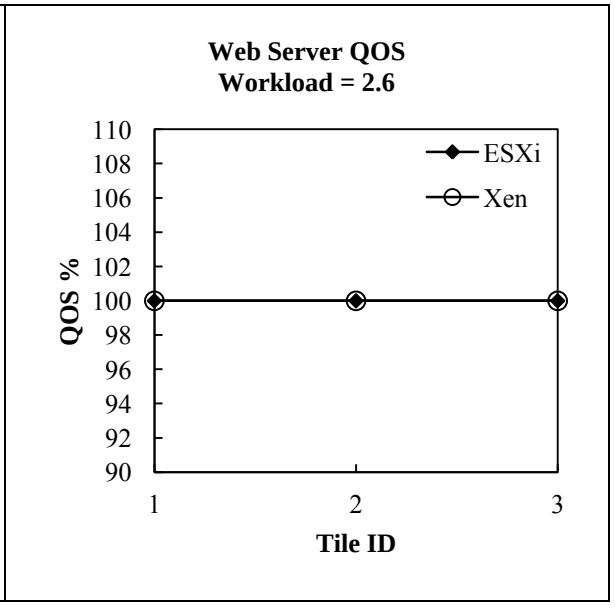

Figure 34: Web Server QOS at 2.6 Workload.

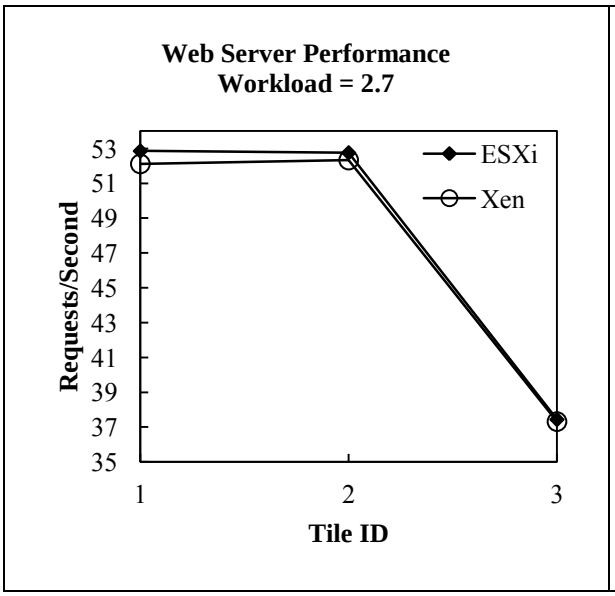

Figure 35: Web Server Performance at 2.7 Workload.

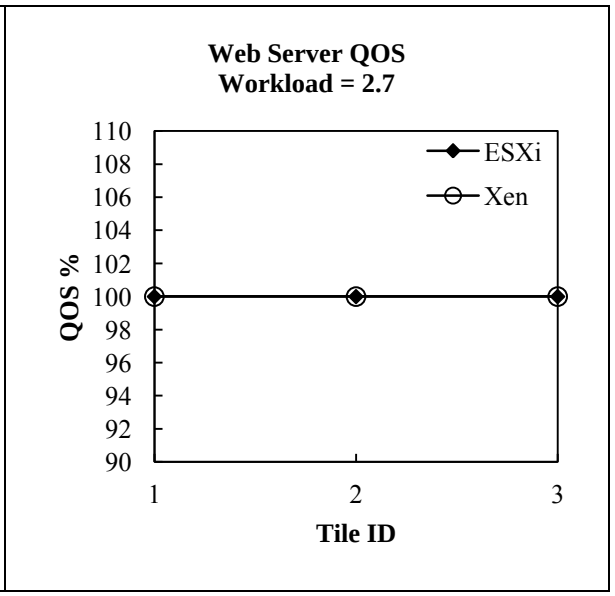

Figure 36: Web Server QOS at 2.7 Workload.

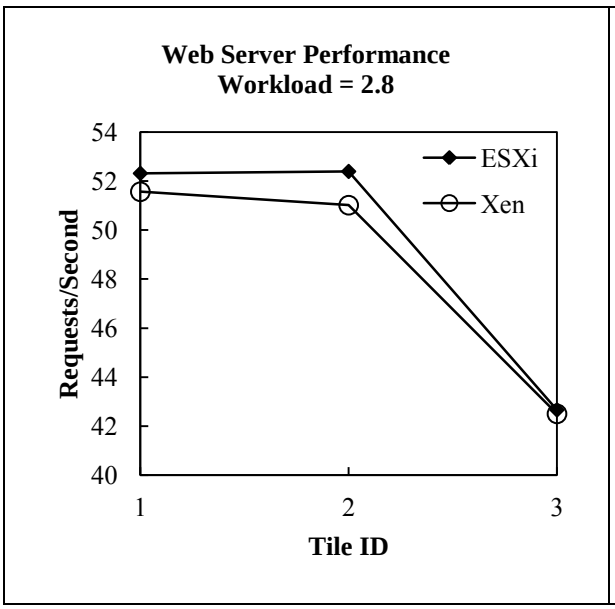

Figure 37: Web Server Performance at 2.8 Workload.

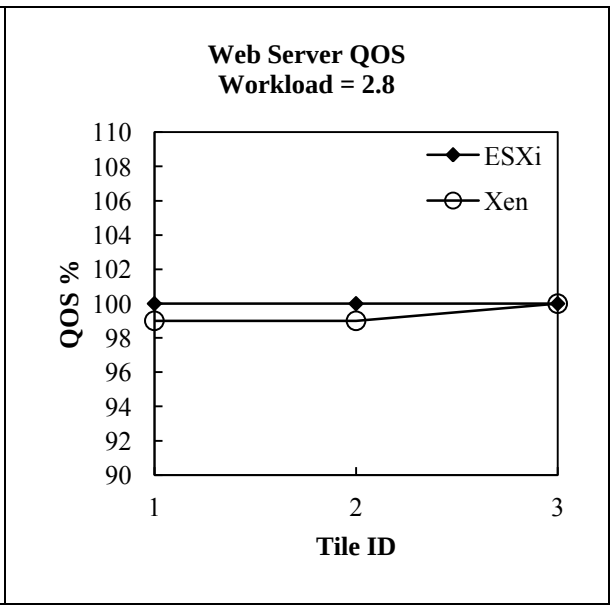

Figure 38: Web Server QOS at 2.8 Workload.

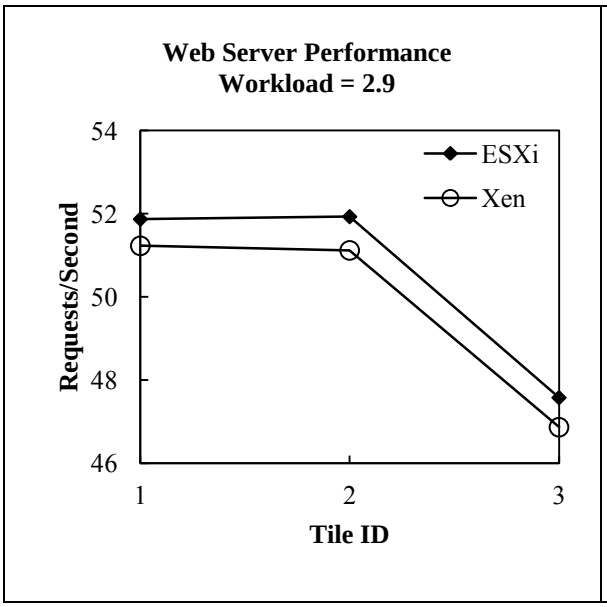

Figure 39: Web Server Performance at 2.9 Workload.

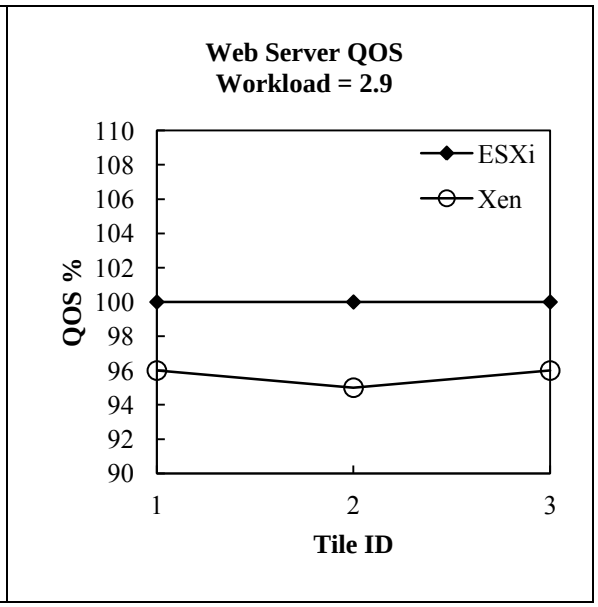

Figure 40: Web Server QOS at 2.9 Workload.

7.1.5 Application Server Performance and QOS by Tile ID

Figure 41 through Figure 54 show the individual application server performance and QOS for workloads ranging from 1.0 to 2.9. Individual application server performance is the best for ESXi at all workloads. Xen‘s application server performance was slightly lower than that of ESXi. KVM‘s application server performance was well below that of ESXi and Xen. ESXi has the best QOS for the whole range of workload. Xen‘s QOS was very good and comparable to that of ESXi for workloads below 2.5.

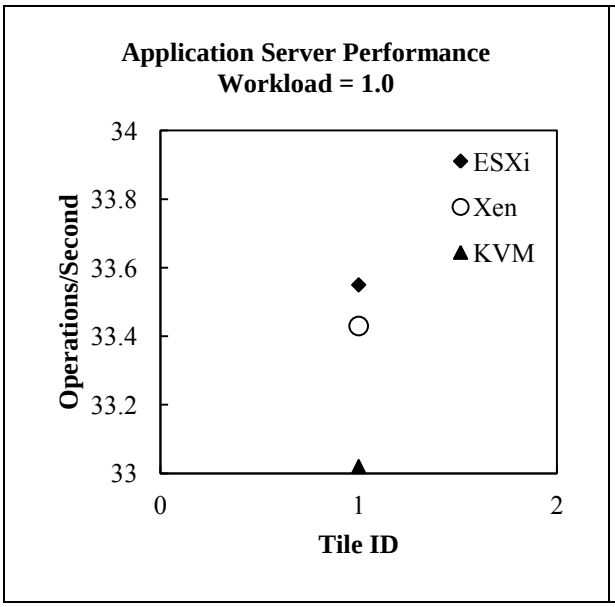

Figure 41: Application Server Performance at 1.0 Workload.

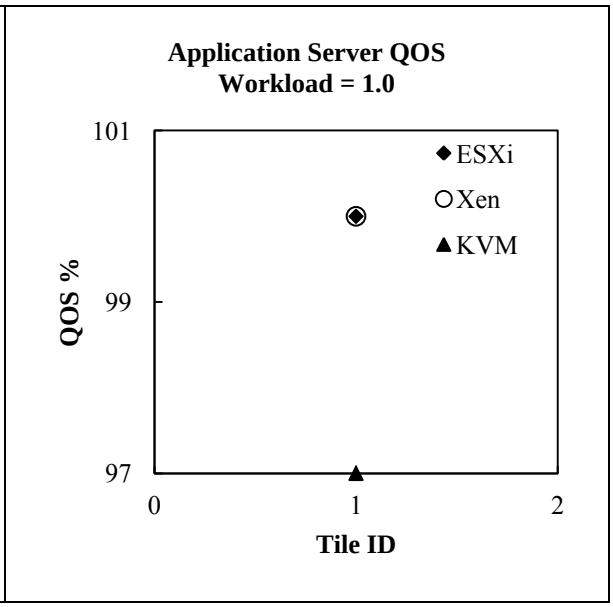

Figure 42: Application Server QOS at 1.0 Workload.

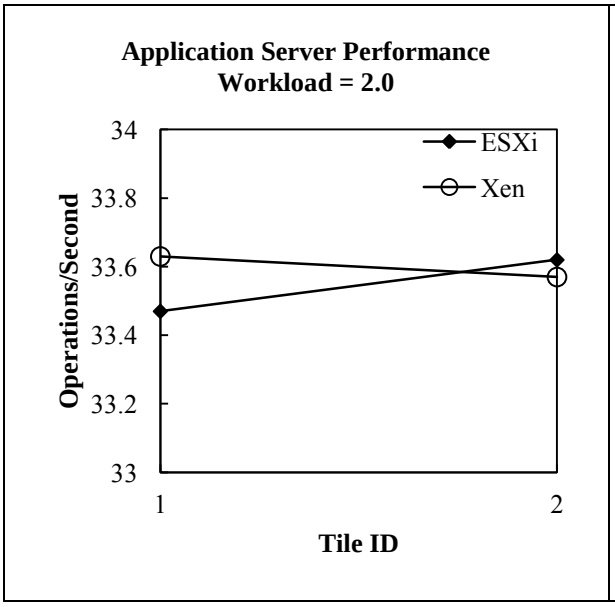

Figure 43: Application Server Performance at 2.0 Workload.

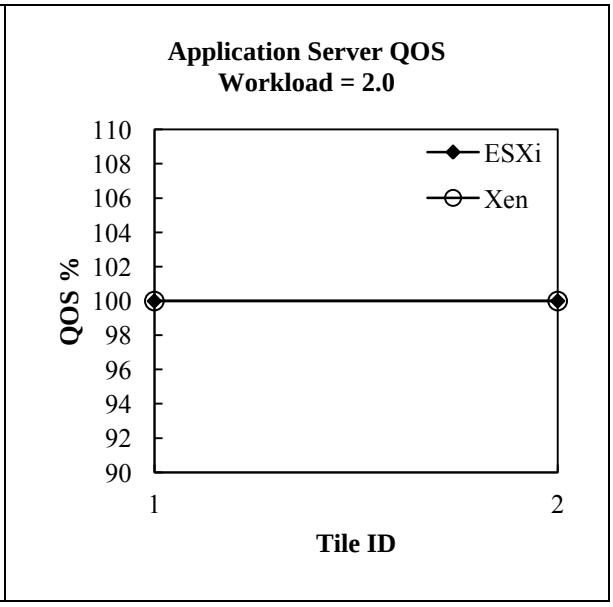

Figure 44: Application Server QOS at 2.0 Workload.

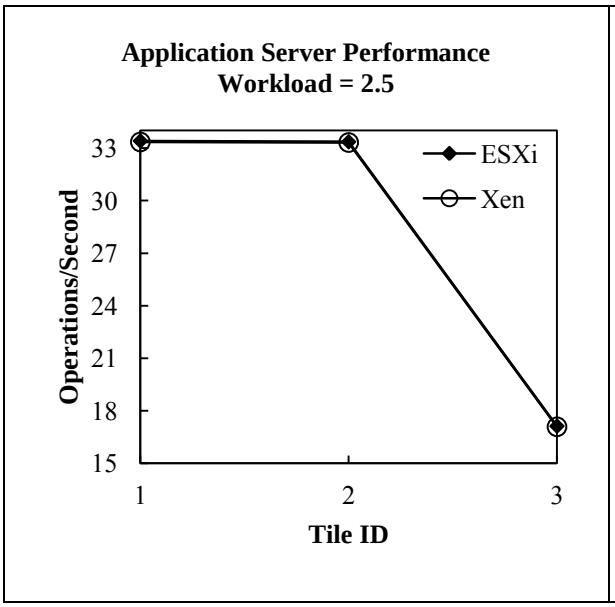

Figure 45: Application Server Performance at 2.5 Workload.

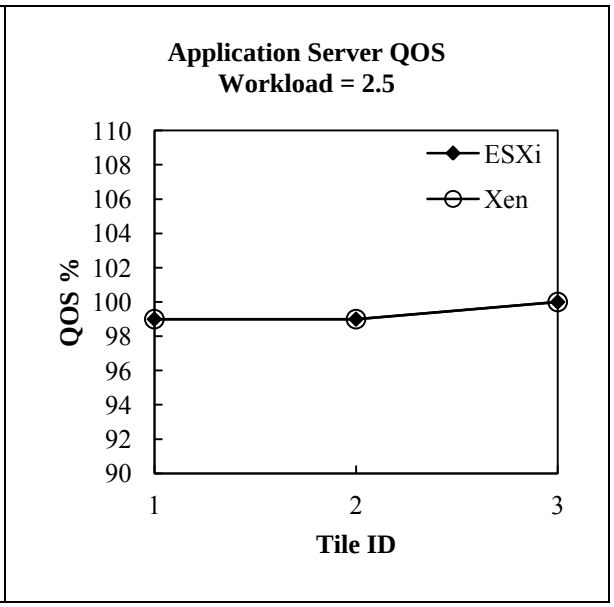

Figure 46: Application Server QOS at 2.5 Workload.

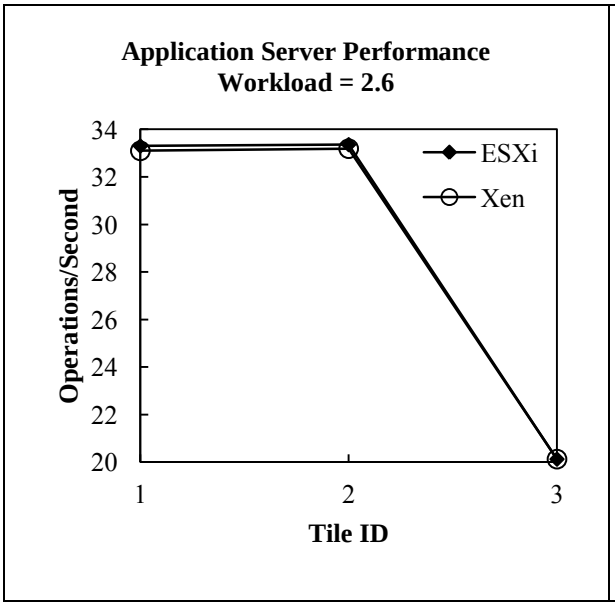

Figure 47: Application Server Performance at 2.6 Workload.

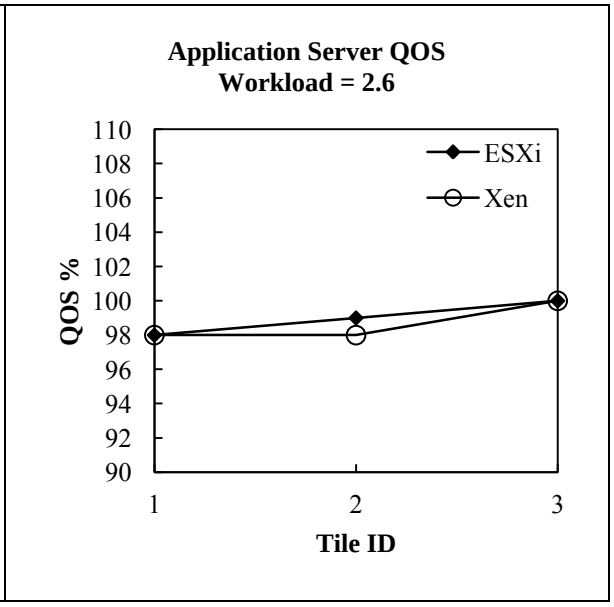

Figure 48: Application Server QOS at 2.6 Workload.

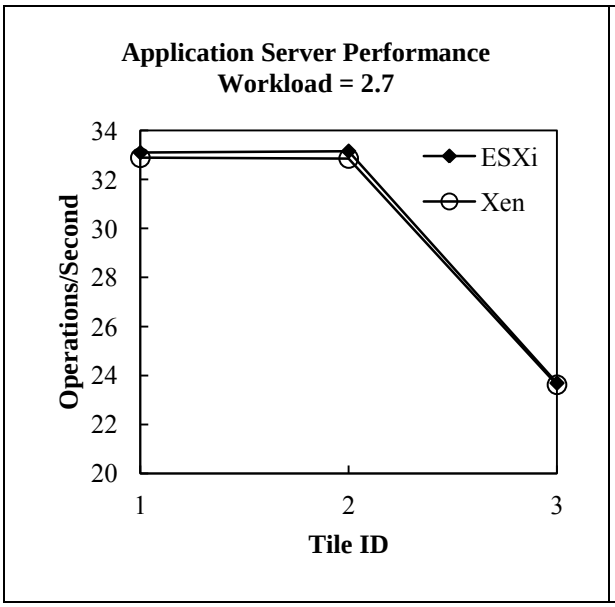

Figure 49: Application Server Performance at 2.7 Workload.

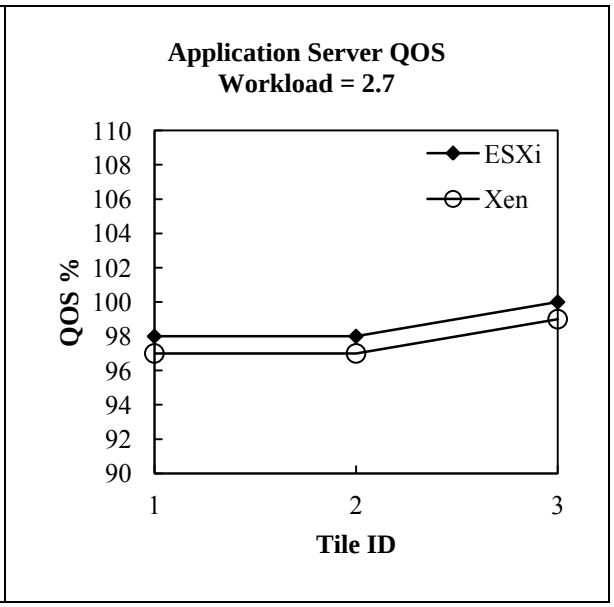

Figure 50: Application Server QOS at 2.7 Workload.

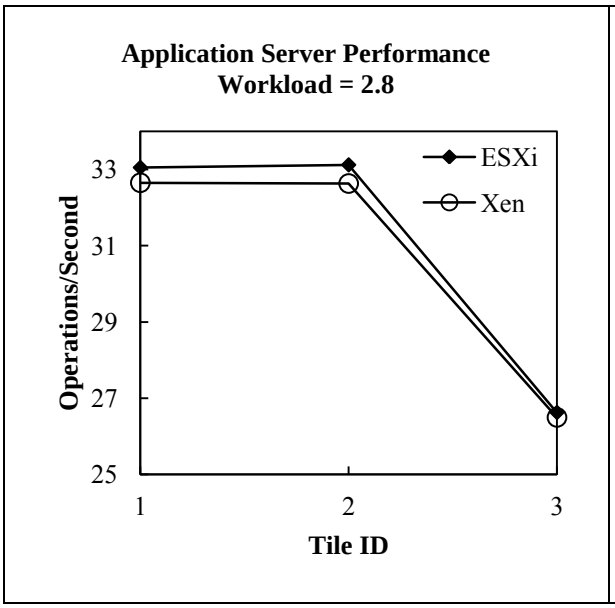

Figure 51: Application Server Performance at 2.8 Workload.

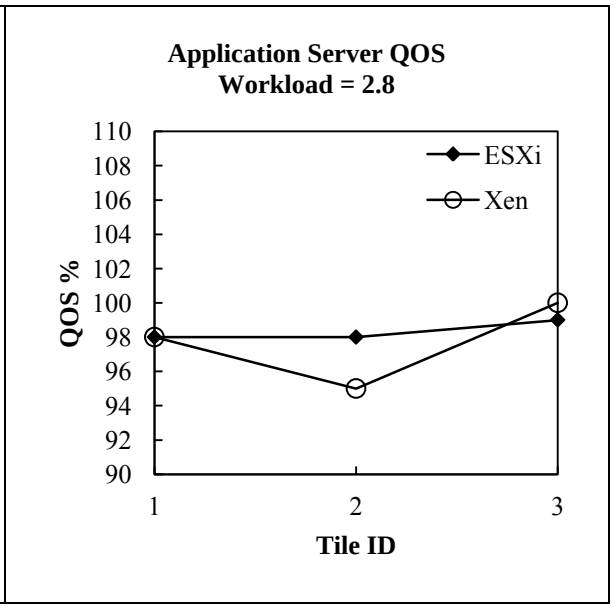

Figure 52: Application Server QOS at 2.8 Workload.

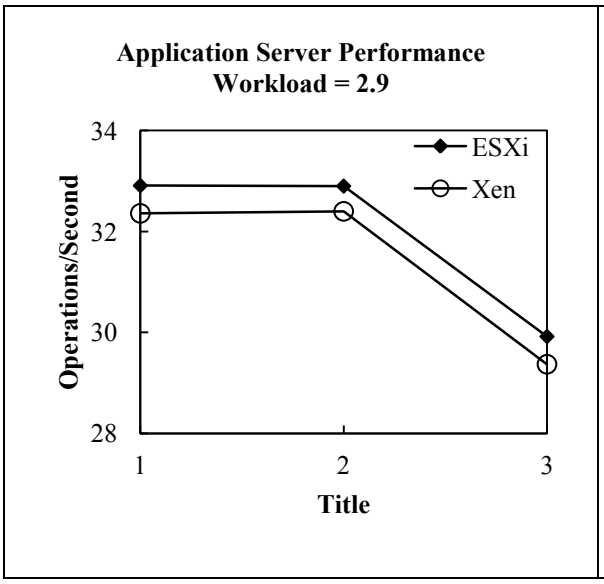

Figure 53: Application Server Performance at 2.9 Workload.

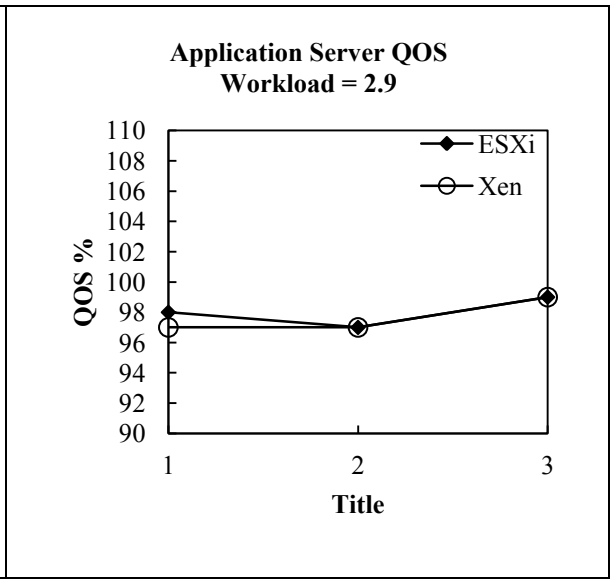

Figure 54: Application Server QOS at 2.9 Workload.

7.1.6 Mail Server Performance and QOS

Figure 55 through Figure 67 shows the individual mail server performance and QOS for workloads ranging from 1.0 to 2.9.

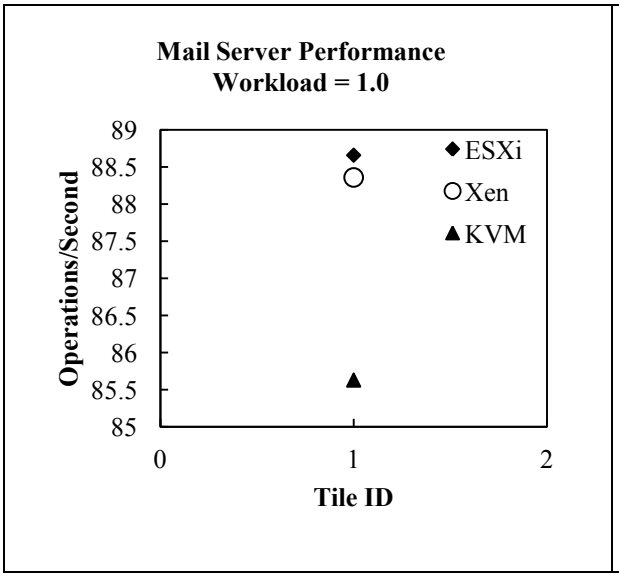

Figure 55: Mail Server Performance at 1.0 Workload.

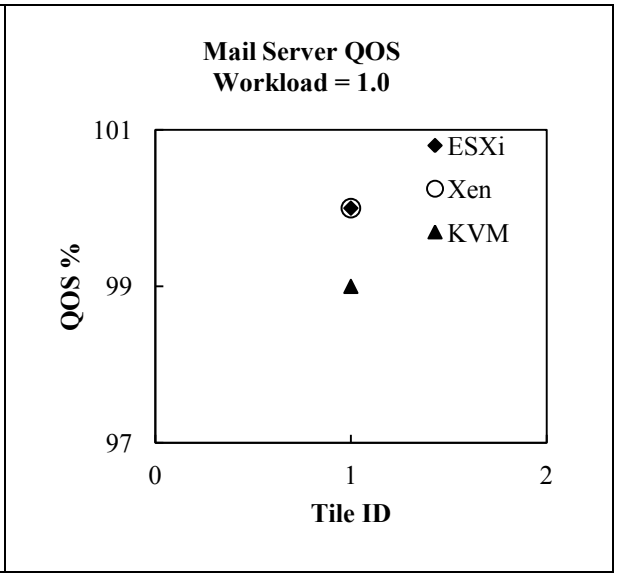

Figure 56: Mail Server QOS at 1.0 Workload.

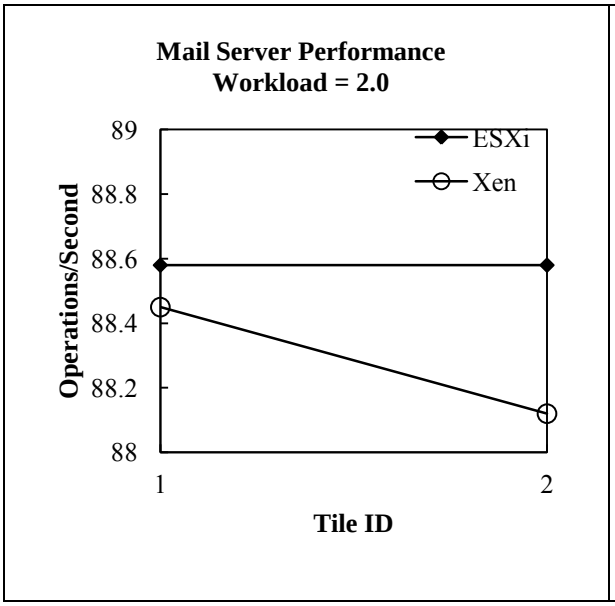

Figure 57: Mail Server Performance at 2.0 Workload.

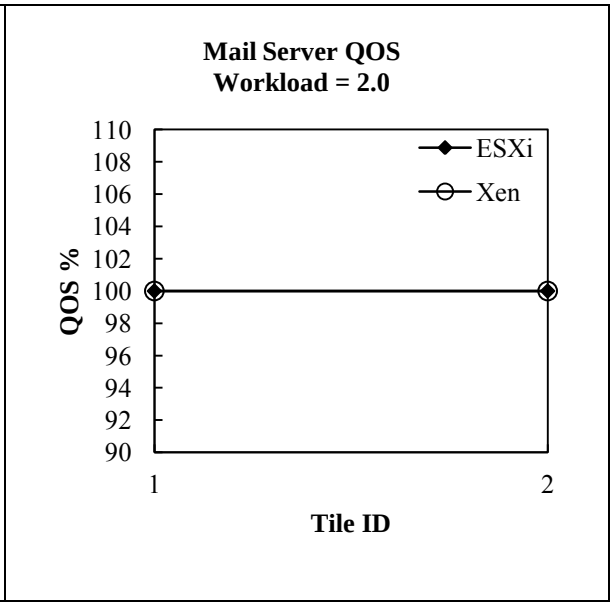

Figure 58: Mail Server QOS at 2.0 Workload.

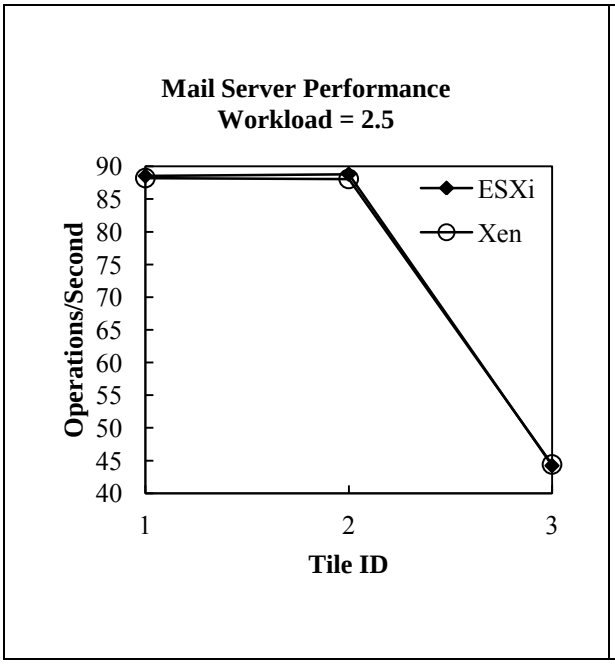

Figure 59: Mail Server Performance at 2.5 Workload.

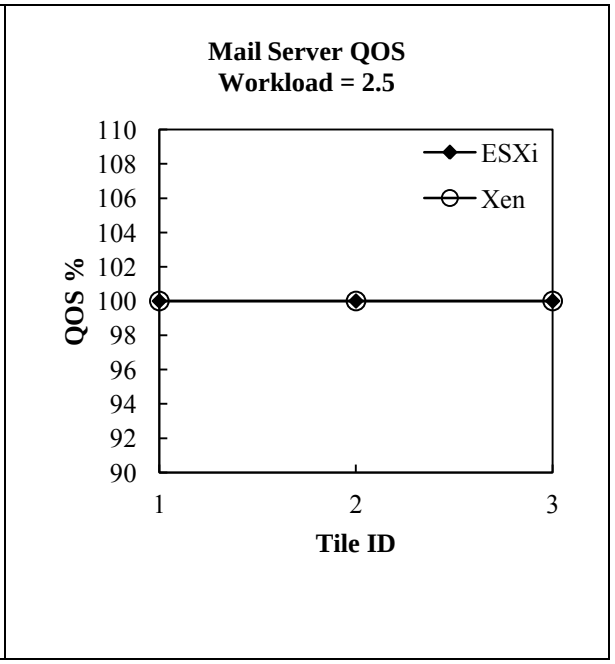

Figure 60: Mail Server QOS at 2.5 Workload.

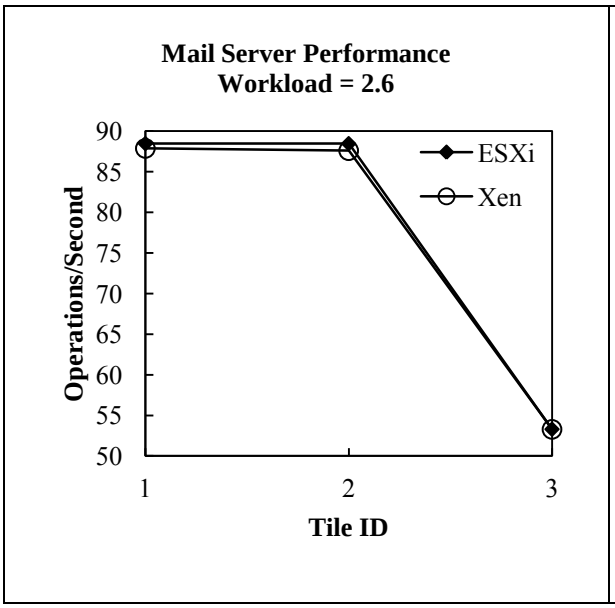

Figure 61: Mail Server Performance at 2.6 Workload.

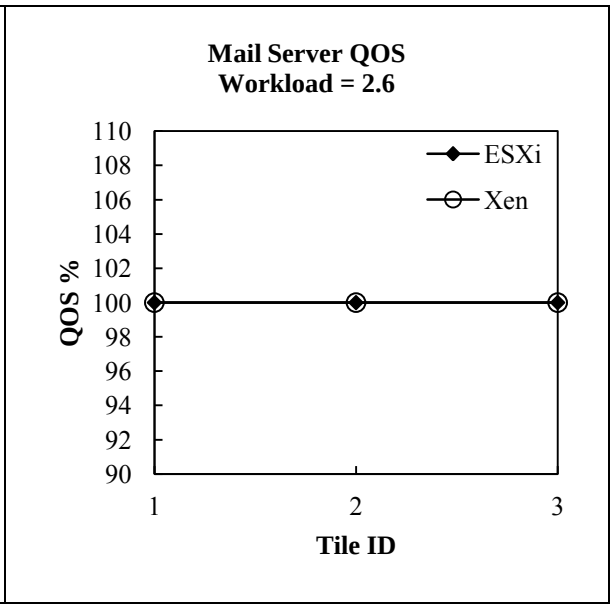

Figure 62: Mail Server QOS at 2.6 Workload.

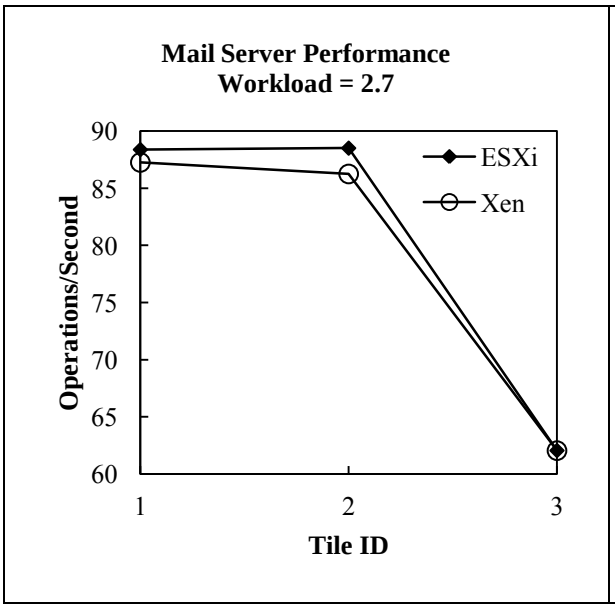

Figure 63: Mail Server Performance at 2.7 Workload.

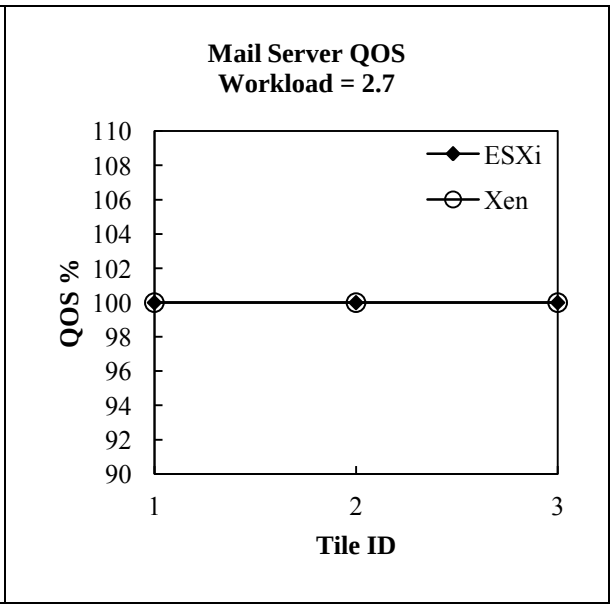

Figure 64: Mail Server QOS at 2.7 Workload.

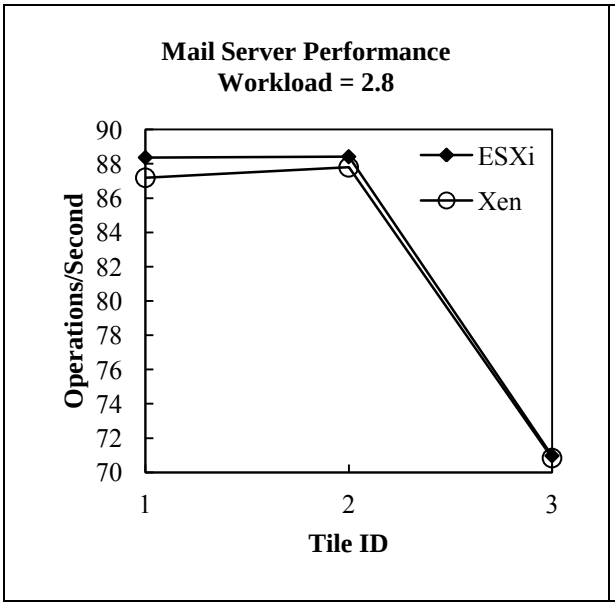

Figure 65: Mail Server Performance at 2.8 Workload.

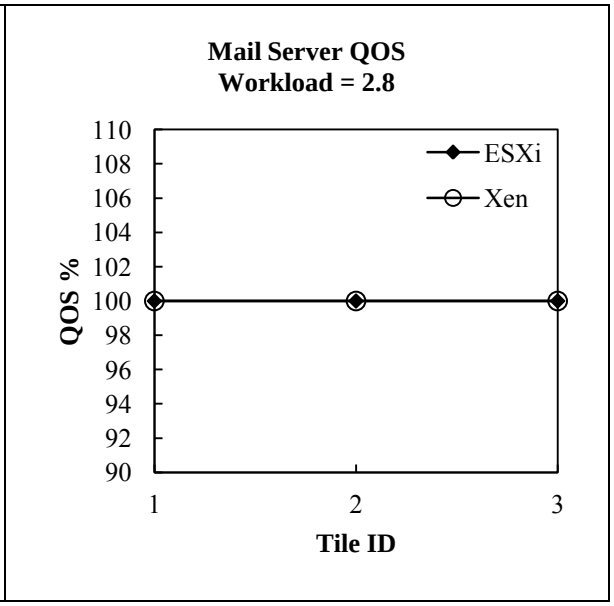

Figure 66: Mail Server QOS at 2.8 Workload.

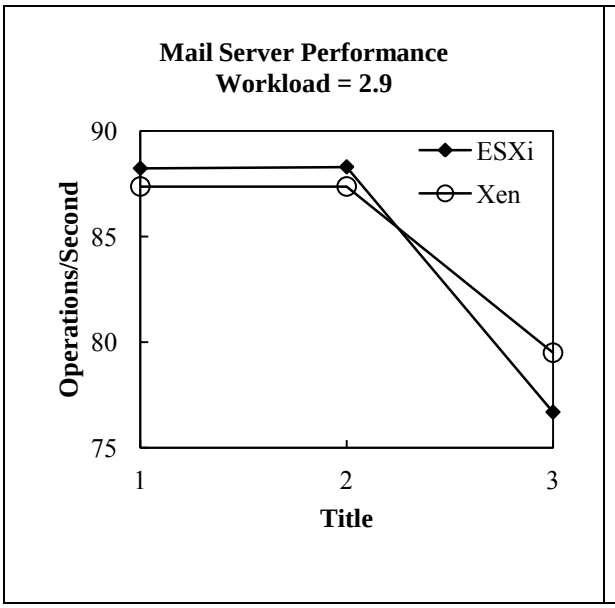

Figure 67: Mail Server Performance at 2.9 Workload.

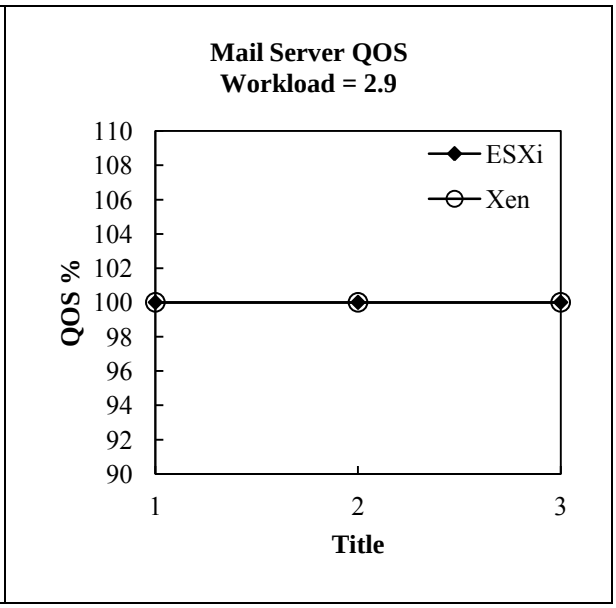

Figure 68: Mail Server QOS at 2.9 Workload.

7.1.7 Statistical Significance Test

Comparison	p-value (Up to 2.6 Tile)	Statistically significant
Overall performance score of ESXi and Xen	0.9816	No
Overall QOS of ESXi and Xen	0.5785	No
Per Tile score of ESXi and Xen	0.9547	No
Web server performance of ESXi and Xen	0.9609	No
AppServer performance of ESXi and Xen	0.9707	No
Mail Server performance of ESXi and Xen	0.9345	No

Table 10: Statistical Significance

Table 10 shows the results of the T-test performed on the result data sets of ESXi and Xen. Since the p-value was greater than 0.05, T-test results show that, the difference in the result data set from ESXi and Xen is not statistically significant. ESXi was performing marginally better than Xen up to 2.6 tiles. For this type of benchmark run where all the conditions were same except the Hypervisor, the results may not be statistically significant using T-test**.**

It should be noted that there is a compliance criterion applied to the performance scores. Performance score by itself cannot be used for comparison of Hypervisors, since, there is an associated compliance pass/fail grade associated for each SPECvirt benchmark run. Xen failed the compliance criteria for workloads of 2.7 and above. Due to the compliance criteria, it might not be appropriate to perform T-test on data sets above 2.7 tiles. Since

there was, only one data point available for KVM, there was no T-test performed using KVM result data set.

7.2 Qualitative Comparison

7.2.1 Installation

ESXi and Xen are bare metal Hypervisors that comes pre-compiled and pre-configured, and thus are very easy to install. KVM, on the other hand, is not a bare metal Hypervisor. KVM is distributed with popular Linux distributions. KVM can be installed by selecting a Linux distribution that has KVM kernel modules. ESXi and Xen being bare metal Hypervisor support a very limited set of hardware configuration. This hardware configuration is specified by the ESXi and Xen manufacturers and is well documented. Since KVM runs on the Linux platform, it can be run from any hardware configuration that the Linux distribution supports. Installing KVM could be very easy on popular distributions like Ubuntu whose official built-in Hypervisor is KVM. When it comes to ease of installation both ESXi and Xen are ahead of KVM.

7.2.2 Management Software

ESXi and Xen come with the management software that could be downloaded from the Hypervisor server using a web browser. ESXi and Xen use their own GUI based management software that uses proprietary communication protocol. There is no other choice of management software available for ESXi and Xen. KVM being an open source Hypervisor comes with a variety of open source management software. Management

software for KVM is available both as a GUI and as terminal application. Even though KVM has many options for management software, they are inferior when it comes to ease of use and features when compared to the proprietary management software of ESXi and Xen.

7.2.3 Hypervisor Administration

Creating, cloning, and deleting, configuration changes of the VM running on ESXi and Xen is very easy using their respective management software. The KVM‘s management GUI used in this work called libvirt can only create and run the virtual machines. For cloning, another terminal based command line utility was used for KVM. For this benchmark, cloning was an important feature that was used to create multi-tiles. Cloning was very easy with ESXi and Xen when compared to KVM.

7.2.4 Guest OS Support

ESXi being a full virtualization Hypervisor supports a variety of operating systems. Many versions of guest operating systems could operate under full performance without any specific modification to the operating system files and with standard hardware drivers. Xen is a para-virtualized Hypervisor and hence supports only a limited set of guest operating systems. For example, Xen does not support the latest version of the Ubuntu as of now. It usually takes many months for Xen to add these new versions of OS to its supported OS list. When an unsupported OS version is run in Xen, there is a considerable performance hit and hence it is not desirable. KVM is also a fully virtualized Hypervisor and hence supports a variety of guest operating systems.

7.2.5 Technology

VMware was founded in 1998 and has been a mature product because it is in the market for a while whereas Xen‘s first release was in 2003. KVM is an open source and is not yet a fully mature product. VMware and KVM Hypervisors provide a completely virtualized set of hardware to the guest operating system. Xen provides high performance drivers only for those OS versions that are supported for the Xen Hypervisor release.

7.2.6 Processor Support

The key hardware that the Hypervisor needs to be virtualized is the processor. The processor is a very complex hardware and its virtualization has a lot of overhead associated. In order to eliminate this overhead caused by processor virtualization Intel and AMD have come up with hardware-assisted virtualization processor technologies called Vtx and AMD-V respectively. Vtx and AMD-V hardware-assisted virtualization technologies that help the Hypervisor to virtualize the processor. Using Vtx and AMD-V, processor instructions executed in each VM are natively executed in the physical processor and hence avoids costly overhead associated with processor virtualization.

Vtx and AMD-V are new technologies that are available only in newer hardware. ESXi and Xen both require Vtx or AMD-V in order to work. ESXi and Xen cannot be run on older hard ware that does not have Vtx or AMD-V supported processor. KVM has the capability to fully virtualize the processor or use the Vtx or AMD-V Supported processor

if it is available. Thus, KVM can be run on any hardware with or without Vtx or AMD-V support.

7.2.7 Usage

Amazon and Rackspace both use Xen, which is the most common Hypervisor. Xen is available from Citrix and other open source solution. Xen and KVM are both favored by the open source communities, with Xen probably the best known (because of Amazon), but KVM getting the most adoption in new Linux deployments. In commercial terms, VMware is the clear winner. For example, Cloudburst supports VMware ESX and ESXi Hypervisors.

Chapter 8

CONCLUSIONS

In this thesis work, three different Hypervisors VMware ESXi, Xen and KVM were benchmarked using SPECvirt_sc2010. Selection of the server hardware was given careful consideration in order to make sure that all three Hypervisors would run without any performance issues due to hardware incompatibility. First-time configuration of SPECvirt was challenging due to multiple changes required to the SPECvirt configuration file. The database server, application server, infrastructure server, web server, idle server and mail server were successfully configured on each of the virtual machine running in a tile.

The benchmark was run at various workloads consisting of single tile, multi-tile and partial tile workloads. The results from the benchmark were used to obtain the point of saturation of the workload for each of the Hypervisors under test. Also overall and individual performance score and the QOS were obtained for each of the Hypervisor under test.

Based on the results it is evident that ESXi's performance is the best, closely followed by Xen. ESXi was able to run the SPECvirt_sc 2010 benchmark with compliance up to 2.8 tile workload, whereas Xen was able to run the benchmark only up to 2.6 tile workload with compliance to SPECvirt_sc2010. ESXi is able to run 6.7% workload more than that of Xen. When using ESXi for large-scale deployments, a 6.7% workload could translate to a significant cost saving on initial hardware purchase as well as operating costs.

T-test results show that the difference in the result data set from ESXi and Xen is not statistically significant. Due to a compliance criteria applied on the performance scores, T-test may not be an appropriate test to compare the data sets since Xen failed SPEC compliance tests for workloads of 2.7 and above.

KVM was the least performing compared to Xen and ESXi. The superior performance of ESXi and Xen could be attributed to the fact that both are bare metal Hypervisors. It is a little bit surprising that Xen Hypervisor, which uses Para-virtualization, was not able to outperform ESXi, which uses full-virtualization.

The poor performance of KVM may be attributed to not being a fully developed product. KVM is relatively new compared to VMware ESXi and Xen and hence may not be fully optimized by the open source development community. When Xen and ESXi are compared, ESXi outperformed Xen marginally. This is a surprise, since Xen advocates always cite the fact that Xen‘s para-virtualized drivers do not have the overhead when compared to the full-virtualized drivers and hence should perform better. However, based on the quantitative performance comparison ESXi outperformed Xen marginally, which is undermining the basic performance advantage of para-virtualized Xen Hypervisor. It seems that the drivers used by the Hypervisors do not contribute that much to the overall performance of the Hypervisor. *The overall performance of the Hypervisor may be highly dependent on the algorithms, optimizations, maturity, scalability and the coding strategy used for the Hypervisor. This could be the reason that ESXi was able to outperform Xen.*

Chapter 9

FUTURE WORK

Cloud Computing makes resources available on-demand from the customer. Virtualization plays a critical role in cloud computing. Two important aspects of virtualization that enable cloud computing are server consolidation and live migration. Server consolidation replaces many servers by virtual servers in one physical server. Live migration is the ability to move virtual machines across many physical servers.

Today‘s standard benchmark SPECvirt_sc2010 presents a fixed load during measurement interval and VMs are in one server. In order to reflect the cloud-computing scenario there is a further need to vary the load during measurement interval so that as in the real world, a virtualized host has to deal with the challenge of managing resources across VMs with varying demands. For this TPC-V, benchmark is being developed as a standard by a standard committee group. As there is more demand for database virtualization instead of diverse workloads, database centric workloads are only aimed at transaction processing or decision support applications [Sethuraman10], and during the measurement interval, the loads are varied. If a Hypervisor is able to meet the TPC-V requirements on multiple server nodes, then the ability of live migration between hosts will also be highlighted by TPC-V.

In this way resource management across many physical servers as per the needs of the user is also studied which characterizes the cloud scenario. The benchmark requires moderate number of virtual machines exercising enterprise applications. This benchmark

is based on TPC-E but cannot be compared to any other TPC-E benchmarks results [Sethuraman10].

9.1 TPC –V Design considerations

To facilitate the creation and loading of many different database sizes in one SUT, and to route different transactions to different Virtual Machines, some properties of TPC-E SUT are modified, but it retains the basic 33 schema and 10 transaction tables of TPC-E SUT. TPC-E SUT is the base for the TPC-V benchmark.

The standard working group has defined three Virtual Machines that together form a Set for the TPC-V benchmark. Tier A component is one virtual machine and the Tier B component of the TPC-E SUT has been divided into two separate Virtual Machines. One virtual machine will handle the Trade-Lookup and Trade-Update transactions, simulating the high storage I/O load of a decision support environment. The second virtual machine will handle all other transactions, which have a CPU-heavy profile and represent an online transaction processing environments. [Sethuraman10].

9.1.1 The Set Architecture

The Set architecture focuses on the following two key areas:

1) Basing the Load on the Performance of the Server: In order to avoid the limitations described in the existing benchmarks, the standard working group has devised a Set [Sethuraman10] architecture where both the number of Sets and the loads placed on each Set increases as the performance of the system increases. The advantage here is that the benchmark will emulate the behavior of real-world servers. Powerful hosts generate more

virtual machines and the virtual machines can handle more load. This ensures that TPC-V is a fitting benchmark for servers of all sizes, and it will stay relevant in the future as servers become more powerful. It is scalable and applicable to all kinds of powerful servers in the future. [Sethuraman10]

2) Varying the load across Sets: In the existing benchmarks, there is a shortcoming that the same exact load is placed on all tiles (or Virtual Machines). In the real world, a virtualized host has to deal with the challenge of managing resources across virtual machines with varying demands. Therefore, each Set in a TPC-V configuration will contribute a different percentage of the overall throughput [Sethuraman10].

The exact number of Sets and the percentage contributed by each Set will depend on the prototyping experiments in the coming year. Metric for TPC-V is assumed in terms of transactions per second, and it is abbreviated to tpsV (the exact benchmark metric is yet to be named and defined) [Sethuraman10]. The following are the numerical values that will be used to initiate the prototyping process:

- A Base Set, which contributes 15% of the overall throughput of the SUT
- A Large Set, which contributes 45% of the overall throughput of the SUT
- Variable Sets contribute the remaining 40% of the overall throughput

Based on the performance of SUT, the exact number of Variable Sets and the division of the 40% among them is calculated. In ―steady state", the performance benchmarks are

measured, where the flow of work requests is adjusted to meet the capabilities of the system in a business model [Sethuraman10].

The peak workload demands for each application are not simultaneous and may not be the same. One workload may be at a peak when another one is low. In such a situation, it enables the computer resources to be shifted from the low-use application to the high-use applications for some period of time, and then shifting the resources to another high demand application at a subsequent point and the process continues. [Sethuraman10].

The dynamic nature of each workload can be affected by a variety of influences that can result in an unpredictable shifting of resources resulting in an equally unpredictable amount of overall system output. Dynamically allocating resources to the virtual machines that are in high demand is a primary requirement of virtualized environment [Sethuraman10]. For any future work on comparison of Hypervisors for cloud environment, TPC-V benchmark may be used considering all the facts discussed above that it benchmarks live migration of workload among virtual machines and varying the load dynamically during the measurement interval similar to a real cloud environment.

REFERENCES

Print Publications

[Barham03]
Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," Proceedings of the Nineteenth ACM Symposium on Operating systems Principles. ACM Press, New York, 2003, pp. 164–177.

[Che08]
Che, J., Q. He, Q. Gao, D. Huang, "Performance Measuring and Comparing of Virtual Machine Monitors," College of Computer Science, Zhejiang University, Hangzhou 310027, China, IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008.

[Fujitsu10A]
FUJITSU, "Benchmark Overview-vServCon," white paper, March 2010.

[Hostway11]
Hostway UK VMware ESXi Cloud Simplified, http://www.hostway.co.uk/small-business/dedicated-hosting/cloud/vmware-esxi.php Comprehensive explanation of the features and benefits of VMware ESXi Hypervisor.

[Makhija06]
Makhija, V., B. Herndon, P. Smith, L. Roderick, E. Zamost, J. Anderson, "VMmark – A Scalable Benchmark for Virtualized systems," Technical Report VMware-TR-2006-002, September 25, 2006.

[Moller07]
Moller, K.T., "Virtual Machine Benchmarking," Diploma Thesis, Karlsruhe Institute of Technology, 2007.

[Nanda05]
Nanda, S., T. Chiueh, "A Survey on Virtualization Technologies," Technical report, Department of Computer Science, SUNY at Stony Brook, New York, 11794-4400,2005.

[Sethuraman10]
Sethuraman, P., H. R. Taheri, TPC-V: A Benchmark for Evaluating the Performance of Database Applications in Virtual Environments. In Proceedings of the Second TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems (TPCTC'10), (2010), Raghunath Nambiar and Meikel Poess (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 121-135.

[Vmware06]
VMware, "Overview of virtualization," white paper, 2006.

[Vmware07B]
VMware, "The Architecture of VMware ESXi," white paper, 2007.

[Xu08]
Xu, X., F. Zhou, J. W. Y. Jiang, "Quantifying Performance Properties of Virtual Machines," School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China, International Symposium on Information Science and Engineering, 2008.

Electronic sources

[Fujitsu10B]
Fujitsu Technology Solutions, "DataSheet Citrix XenServer," http://sp.ts.fujitsu.com/dmsp/Publications/public/ds-XenServer.pdf, last accessed January 06, 2010.

[Spec11]
Standard Performance Evaluation Corporation, "SPECvirt_sc2010 Design Overview," http://www.spec.org/virt_sc2010/, last accessed May 23, 2011.

[Vmware07A]
VMware, "A Performance Comparison of Hypervisors," http://www.vmware.com/pdf/Hypervisor_performance.pdf, last accessed January 31, 2007.

[Xen09]
Xen, "How does Xen work,"http://www.xen.org/files/Marketing/HowDoesXenWork.pdf, December 2009.

[Xen Source07]
XenSource, "A Performance Comparison of Commercial Hypervisors," http://www.cc.iitd.ernet.in/misc/cloud/XenExpress.pdf, 2007.

APPENDIX A

Besim Output

```
Testing BESIM Requests for Ecommerce Workload



http://infraserver:81/besim.dll?3&0&1079975569&500

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&0&1079975569&500
<pre>
0
DONE ResetDate = 20111113, Time=1079975569,Load=500,SL=11
</pre>
</body></html>



http://infraserver:81/besim.dll?3&1

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&1
<pre>
0
5569&1200MHz Computers
5570&1200MHz Computers
5571&1200MHz Computers
5572&1200MHz Computers
5573&1200MHz Computers
5574&1200MHz Computers
5575&1200MHz Computers
</pre>
</body></html>



http://infraserver:81/besim.dll?3&2&5

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
```

```
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&2&5
<pre>
0
0569&Computers PRO0000
0570&Computers PRO0000
0571&Computers PRO0000
0572&Computers PRO0000
0573&Computers PRO0000
0574&Computers PRO0000
0575&Computers PRO0000
0576&Computers PRO0000
0577&Computers PRO0000
0578&Computers PRO0000
0579&Computers PRO0000
0580&Computers PRO0000
0581&Computers PRO0000
</pre>
</body></html>
```

```
http://infraserver:81/besim.dll?3&3&Pro+Home+PDA

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&3&Pro+Home+PDA
<pre>
0
5569&Computers PRO0000
5570&Computers PRO0000
5571&Computers PRO0000
5572&Computers PRO0000
5573&Computers PRO0000
5574&Computers PRO0000
</pre>
</body></html>
```

```
http://infraserver:81/besim.dll?3&4&500

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&4&500
<pre>
0
All
Application
```

```
Audio Drivers
BIOS
Chipset
Communication Drivers
Diagnostics
IDE/SCSI
Input Drivers
Keyboard  Drivers
Monitors
Network Drivers
Patches
Removable Media Drivers
Security Patches
Software Dev. Tools
System Utilities
System Management
Video Drivers
Virus Protection
</pre>
</body></html>
```

```
http://infraserver:81/besim.dll?3&5

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&5
<pre>
0
Arabic
Bulgarian
Chinese-S
Chinese-T
Czech
Danish
Dutch
English
Estonian
Finnish
French
German
Greek
Hebrew
Hungarian
Indonesian
Italian
Japanese
Korean
Norwegian
Pan-Euro
Polish
Portuguese
```

```
Russian
Slovak
Slovenian
Spanish
Swedish
Thai
Turkish
</pre>
</body></html>
```

```
http://infraserver:81/besim.dll?3&6&200

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&6&200
<pre>
0
RNZX
RN3K
RN2045
Enterprise Xilin V3.01
Desktop Xilin V3.0
OS 743 for Architecture N7
NP-OS V13.41
CafeOS 2.3.1 for HA82
CafeOS 2.3.1 for NA90
FreeBinOS 7.5
</pre>
</body></html>
```

```
http://infraserver:81/besim.dll?3&7&2000&Desktops&Dutch&RNZX

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&7&2000&Desktops&Dutch&RNZX
<pre>
0
020000&RNZX_BIOS.exe&2004-6-1 10:15am&138000&This is the executable
binary for the PRO0000 Personal Computers using RNZX_BIOS
020001&RNZX_BIOS.exe&2004-6-1 10:15am&140000&This is the executable
binary for the PRO0000 Personal Computers using RNZX_BIOS
020002&RNZX_BIOS.exe&2004-6-1 10:15am&142000&This is the executable
binary for the PRO0000 Personal Computers using RNZX_BIOS
020003&RNZX_BIOS.exe&2004-6-1 10:15am&144000&This is the executable
binary for the PRO0000 Personal Computers using RNZX_BIOS
```

```
</pre>
</body></html>



http://infraserver:81/besim.dll?3&8&12345

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&8&12345
<pre>
0
123450&RNZX_BIOS.exe&2004-6-1 10:15am&7024305&http://www.SPECweb2005-
Ecommerce.web/Computers/computer.script?1200MHz:new
This is the executable binary for the PRO0000 Personal Computers using
RNZX_BIOS
1. Click the new computer click to the new computer.<BR>2. Click the
new computer click to the new computer.<BR>3. Click the new computer
click to the new computer.<BR>4. Click the new computer click to the
new computer.<BR>5. Click the new computer click to the new
computer.<BR>6. Click the new computer click to the new computer.<BR>7.
Click the new computer click to the new computer.<BR>8. Click the new
computer click to the new computer.<BR>9. Click the new computer click
to the new computer.<BR>10. Click the new computer click to the new
computer.<BR>11. Click the new computer click to the new
computer.<BR>12. Click the new computer click to the new
computer.<BR>13. Click the new computer click to the new
computer.<BR>14. Click the new computer click to the new
computer.<BR>15. Click the new computer click to the new
computer.<BR>16. Click the new computer click to the new computer.<BR>
      The new computer click fast.  The new computer click fast. The
computer is fast. Please Reboot Now.....................
</pre>
</body></html>



http://infraserver:81/besim.dll?3&0&1079978064&1234

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&0&1079978064&1234
<pre>
0
DONE ResetDate = 20111113, Time=1079978064,Load=1234,SL=17
</pre>
</body></html>
```

```
http://infraserver:81/besim.dll?3&8&1009

<html>
<head><title>SPECweb2005 BESIM</title></head>
<body>
<p>SERVER_SOFTWARE = Microsoft-IIS/7.5
<p>REMOTE_ADDR = 1.1.1.1
<p>SCRIPT_NAME = /besim.dll
<p>QUERY_STRING = 3&8&1009
<pre>
0
010090&RNZX_BIOS.exe&2004-6-1 10:15am&64576&http://www.SPECweb2005-
Ecommerce.web/Computers/computer.script?1200MHz:new
This is the executable binary for the PRO0000 Personal Computers using
RNZX_BIOS
1. Click the new computer click to the new computer.<BR>2. Click the
new computer click to the new computer.<BR>3. Click the new computer
click to the new computer.<BR>4. Click the new computer click to the
new computer.<BR>5. Click the new computer click to the new
computer.<BR>6. Click the new computer click to the new computer.<BR>7.
Click the new computer click to the new computer.<BR>8. Click the new
computer click to the new computer.<BR>9. Click the new computer click
to the new computer.<BR>10. Click the new computer click to the new
computer.<BR>11. Click the new computer click to the new
computer.<BR>12. Click the new computer click to the new
computer.<BR>13. Click the new computer click to the new
computer.<BR>14. Click the new computer click to the new
computer.<BR>15. Click the new computer click to the new
computer.<BR>16. Click the new computer click to the new computer.<BR>
      The new computer click fast.  The new computer click fast. The
computer is fast. Please Reboot Now.....................
</pre>
</body></html>
```

APPENDIX B

SPECvirt Results Output Sample

SPECvirt_sc2010 Result

Custom Built: P7P55D-E ASUS Mother Board VMWare: ESXi 4.1	**SPECvirt_sc2010 262.6 @ 18 VMs**

Tested By: University of North Florida	**SPEC License #:** 4120	**Test Date:** Sep-2011

Performance Section Performance Summary Performance Details Validation Errors	**SUT Configuration Section** Physical Configuration Virtual Configuration	**Power Section** N/A	**Notes Section** Physical System Notes Virtualization SW Notes Hosted VM Notes

Performance Summary:

Performance							
Tile #	**Pct Load**	**Application Server**	**Web Server**	**Mail Server**	**Idle Server**	**Per-Tile Score**	**Overall Score**
1	100%	33.10	52.86	88.39	N/A	96.94	262.6
2	100%	33.16	52.76	88.52	N/A	96.98	
3	70%	23.69	37.44	62.06	N/A	68.69	

Quality Of Service (QOS)							
Tile #	**Pct Load**	**Application Server**	**Web Server**	**Mail Server**	**Idle Server**	**Per-Tile Score**	**Overall Score**
1	100%	0.98	1.00	1.00	1.00	1.00	99.63%
2	100%	0.98	1.00	1.00	1.00	1.00	
3	70%	1.00	1.00	1.00	1.00	1.00	

Performance Details:

Tile 1

Application Server					
Req. Type	Req/sec	Avg Resp. Time	Max Resp. Time	90th%	Required 90th%
Manufacturing	13.05	1.80	12.15	3.25	5
Dealer	20.06	0.19/0.21/0.22	2.17/3.66/1.96	0.40/0.50/0.50	2/2/2

Web Server					
Web Wkload	Req/sec	Good	Tolerable	Fail	Valid. Errors
Support	52.86	380412	159	4	2

Mail Server					
Req. Type	Req/sec	Total Count	Pass Count	Fail Count	Pass Pct
Append	14.23	102462	102462	0	100.00
Fetch	36.11	259972	259972	0	100.00

Idle Server				
Req. Type	Total	Avg. Resp. Msec	Min Resp. Msec	Max Resp. Msec
Heartbeats	720	12.12	1	78

Tile 2

Application Server					
Req. Type	Req/sec	Avg Resp. Time	Max Resp. Time	90th%	Required 90th%
Manufacturing	12.99	1.87	14.23	3.75	5
Dealer	20.17	0.20/0.22/0.23	2.13/3.13/1.58	0.50/0.50/0.50	2/2/2

Web Server					
Web Wkload	Req/sec	Good	Tolerable	Fail	Valid. Errors
Support	52.76	379542	293	5	1

Mail Server					
Req. Type	Req/sec	Total Count	Pass Count	Fail Count	Pass Pct
Append	14.23	102451	102451	0	100.00
Fetch	36.15	260301	260301	0	100.00

Idle Server				
Req. Type	Total	Avg. Resp. Msec	Min Resp. Msec	Max Resp. Msec
Heartbeats	720	12.42	1	77

Tile 3

Application Server					
Req. Type	Req/sec	Avg Resp. Time	Max Resp. Time	90th%	Required 90th%
Manufacturing	9.48	1.18	5.44	1.50	5
Dealer	14.21	0.14/0.16/0.16	2.22/2.00/1.39	0.30/0.40/0.40	2/2/2

Web Server					
Web Wkload	Req/sec	Good	Tolerable	Fail	Valid. Errors
Support	37.44	269563	0	0	2

Mail Server					
Req. Type	Req/sec	Total Count	Pass Count	Fail Count	Pass Pct
Append	9.99	71924	71924	0	100.00
Fetch	25.26	181872	181872	0	100.00

Idle Server

Req. Type	Total	Avg. Resp. Msec	Min Resp. Msec	Max Resp. Msec
Heartbeats	720	12.56	1	80

Validation Errors:
No Validation Errors Found

Physical Configuration:

System Under Test (SUT)	
Virt. Vendor/Product	VMWare ESXi 4.1
# of SUTs	1
Server Vendor	Custom Built
Server Model	P7P55D-E ASUS Mother Board
Processor	Intel Core i7-875K
Processor Speed (MHz)	2930
Processor Cores	4 cores, 1 chips, 4 cores/chip, 2 threads/core
Primary Cache	32 KB I + 32 KB D on chip per core
Secondary Cache	256 KB I+D on chip per core
Other Cache	8 MB I+D on chip per chip
Memory	16384 MB SDRAM
Operating System	N/A
File System	ext3
Other Hardware	N/A
Other Software	N/A

SUT Storage	
Storage Controllers	Intel P55 Express Chipset Onboard
Storage Enclosure	N/A
Disk Description	1 x 500GB SATA2 7200RPM 3 x 120GB SATA3 SSD
RAID Level	N/A
UPS Required?	No

SUT Network	
Network Adapters	Intel PWLA8391GT PRO/1000 GT PCI Network Adapter
SUT Ports Total	1
SUT Ports Used	1
Network Type	1 Gigabit Ethernet
Network Speed	1000 Mbps

Clients	
Model	Intel DP67BGB3 Mother Board
# of Clients	3
Processor	Intel Core i7 2600K
Processor Speed (MHz)	3400
# Processors	2
Memory	4096 MB SDRAM
Network Controller	Citrix PV Ethernet Adapter
Operating System	Windows 7 Pro SP1
JVM Version	Java(TM) SE Runtime Environment (build 1.6.0_26-b03)
Other Hardware	Physical Network Card: On board Intel 82579V 1000Mbps
Other Software	XenServer 5.6 Service Pack 2 (Build 47101p) for running Clients VM

Availability Dates	
SUT Hardware	Oct-2010
Virt. Software	April-2011
Other Components	N/A

Virtual Configuration:

Web Server

VM Configuration Details	
# VCPUS	1
VCPU Speed (MHz)	2930
Memory (MB)	800
# VNICs	2
VNIC Description	Intel Pro/1000MT
Storage Description	VMWare Virtual IDE Hardware ATA Device
Virtual Disk Size (MB)	19968
Datastore Size (MB)	N/A
VM OS	Windows Server 2008 R2 Enterprise SP1
VM OS Availability	Oct-2009

Web Server Configuration Details	
Web Server Vendor	Microsoft
Web Server Name/Version	IIS 7
Availability Date	Oct-2009
Script Vendor	www.PHP.Net
Script Name/Version	PHP version 5.3.6
Script Availability Date	Mar-2011
JVM Version	Java(TM) SE Runtime Environment (build 1.6.0_26-b03)
Other Software	Smarty Template Engine 2.6.26

Application Server

VM Configuration Details	
# VCPUS	2
VCPU Speed (MHz)	2930
Memory (MB)	1200
# VNICs	1
VNIC Description	Intel Pro/1000MT
Storage Description	VMWare Virtual IDE Hardware ATA Device
Virtual Disk Size (MB)	19968
Datastore Size (MB)	N/A
VM OS	Windows Server 2008 R2 Enterprise SP1
VM OS Availability	Oct 2009

App Server Configuration Details	
Application Server Vendor	Sun Microsystems
App. Server Name/Version	GlassFish Enterprise Server V2.1.1
Availability Date	Oct-2009
Emulator Vendor	Apache
Emulator Name/Version	Tomcat-7.0.16
Emulator Availability Date	May-2011
JVM Description	Oracle Java(TM) SE Runtime Environment (build 1.6.0_26-b03)
JVM Availability	Jun-1999
Other Software	N/A

Mail Server

VM Configuration Details	
# VCPUS	1
VCPU Speed (MHz)	2930
Memory (MB)	800
# VNICs	1
VNIC Description	Intel Pro/1000MT
Storage Description	VMWare Virtual IDE Hardware ATA Device

Mail Server Configuration Details	
Mail Server Vendor	www.Dovecot.org
Mail Server Name/Version	Dovecot 1.2.15
Availability Date	Oct-2010
JVM Version	OpenJDK Runtime Environment (IcedTea6 1.10.2)
Other Software	N/A

Virtual Disk Size (MB)	20480
Datastore Size (MB)	N/A
VM OS	Ubuntu 11.04
VM OS Availability	Apr-2011

Database Server

VM Configuration Details	
# VCPUS	1
VCPU Speed (MHz)	2930
Memory (MB)	1200
# VNICs	1
VNIC Description	Intel Pro/1000MT
Storage Description	VMWare Virtual IDE Hardware ATA Device
Virtual Disk Size (MB)	22118
Datastore Size (MB)	N/A
VM OS	Windows Server 2008 R2 Enterprise SP1
VM OS Availability	Oct-2009

Database Configuration Details	
Database Vendor	MySQL
Database Name/Version	MySQL 5.5.14
Availability Date	Jul-2011
JVM Version	Java SE Runtime Environment (build 1.6.0_26-b03)
Other Software	N/A

Infraserver

VM Configuration Details	
# VCPUS	1
VCPU Speed (MHz)	2930
Memory (MB)	500
# VNICs	2
VNIC Description	Intel Pro/1000MT
Storage Description	VMWare Virtual IDE Hardware ATA Device
Virtual Disk Size (MB)	50688
Datastore Size (MB)	N/A
VM OS	Windows Server 2008 R2 Enterprise SP1
VM OS Availability	Oct-2009

Infraserver Configuration Details	
Web Server Vendor	Microsoft
Web Server Name/Version	IIS 7
Availability Date	Oct-2009
Script Vendor	IIS
Script Name/Version	ISAPI
Script Availability Date	Oct-2009
JVM Version	Java SE Runtime Environment (build 1.6.0_26-b03)
Other Software	N/A

Idle Server

VM Configuration Details	
# VCPUS	1
VCPU Speed (MHz)	2930
Memory (MB)	256
# VNICs	1
VNIC Description	Intel Pro/1000MT
Storage Description	VMWare Virtual IDE Hardware ATA Device
Virtual Disk Size (MB)	5519
Datastore Size (MB)	N/A
VM OS	Microsoft XP Professional Version 2002 Service Pack 3
VM OS Availability	2002

Idle Server Configuration Details	
JVM Version	Java SE Runtime Environment (build 1.6.0_25-b06)
Other Software	N/A

Notes:
Physical System Notes

Storage Notes
VM of Application Server, Database Server, Mail Server and Infrastructure Server for each tile was stored and run from its own 120GB solid state drive.

Virtualization Software Notes

Web Server VM Notes

Application Server VM Notes

Mail Server VM Notes
File system loaded

Database Server VM Notes

Infraserver VM Notes

Idle Server VM Notes

Client Driver Notes

Other Notes

For questions about this result, please contact the submitter: University of North Florida

APPENDIX C

Client – Master‘s SPECvirt Control.config file Content

```
########################################################################
#
#
# Control.config
#
# SPECvirt_sc2010 properties file.
#
# Copyright (c) 2004-2009 Standard Performance Evaluation Corporation
(SPEC)
# All rights reserved.
#
########################################################################
#


########################################################################
#
# CONFIGURABLE WORKLOAD PROPERTIES
# These values can be modified to suit your needs. However, some
configurable
# properties still have limited ranges of valid values, and compliant
runs must
# conform to these limits. Any such restrictions are specified in the
property
# descriptions above the property name.
########################################################################
#

# NUM_TILES is the number of sets of workloads you intend to run.
Increase this
# value to increase load. However, if you cannot run another complete
tile of
# workloads, consider using the LOAD_SCALE_FACTORS[x] property,
described below
# to run a partially loaded tile.

NUM_TILES = 1

SPECVIRT_HOST = suganya
SPECVIRT_RMI_PORT = 9990

# RMI_TIMEOUT is the number of seconds SPECvirt will wait for the prime
clients
# to start their RMI servers before aborting the benchmark run

RMI_TIMEOUT = 30

# Use TILE_ORDINAL to control which sets of PRIME_HOST clients to use
for the
```

```
# run. The value specified corresponds to the "tile" number index
specified in
# the PRIME_HOSTS key (i.e. PRIME_HOSTS[tile][workload]. If commented
out,
# then the benchmark will start with PRIME_HOST[0][workload] and
increment the
# PRIME_HOST tile index until NUM_TILES is reached. If used, then the
# TILE_ORDINAL index and value for *all* tiles must be specified
(starting
# with 0).
TILE_ORDINAL[0] =0
TILE_ORDINAL[1] =1
TILE_ORDINAL[2] =2


# PRIME_HOST specifies the hostname and port number for each prime
client
# (or workload controller). The indexes used specify the tile and
workload ID
# and therefore must be unique. If there are multiple prime clients on
a single
# host, then each must listen on a different port number. There will be
one
# PRIME_HOST per workload, "NUM_WORKLOADS" PRIME_HOSTs per TILE
(default: 4).
# The format is PRIME_HOST[tile][workload] = "<host>:<port>" where the
values
# for the "workload"  and "tile" indexes are never greater than
# NUM_WORKLOADS - 1 and NUM_TILES - 1, respectively.

PRIME_HOST[0][0] = "suganya1:1098"
PRIME_HOST[0][1] = "suganya1:1096"
PRIME_HOST[0][2] = "suganya1:1094"
PRIME_HOST[0][3] = "suganya1:1092"

PRIME_HOST[1][0] = "suganya2:1098"
PRIME_HOST[1][1] = "suganya2:1096"
PRIME_HOST[1][2] = "suganya2:1094"
PRIME_HOST[1][3] = "suganya2:1092"

PRIME_HOST[2][0] = "suganya3:1098"
PRIME_HOST[2][1] = "suganya3:1096"
PRIME_HOST[2][2] = "suganya3:1094"
PRIME_HOST[2][3] = "suganya3:1092"


# PRIME_HOST[1][0] = "127.0.1.1:1078"
# PRIME_HOST[1][1] = "127.0.1.1:1076"
# PRIME_HOST[1][2] = "127.0.1.1:1074"
# PRIME_HOST[1][3] = "127.0.1.1:1072"

# SPECVIRT_INIT_SCRIPT and SPECVIRT_EXIT_SCRIPT are used to run a
single
# script on the prime controller before and/or after a benchmark run.
Likewise,
# PRIME_HOST_INIT_SCRIPT and PRIME_HOST_EXIT_SCRIPT are used to run
scripts on
```

```
# the prime client systems before and/or after a benchmark run. If a
path is
# included with the script name, it must be the full path. Specifying a
file
# name only assumes the file exists in the current working directory of
the
# prime client (typically the location of clientmgr.jar)

# SPECVIRT_INIT_SCRIPT = "init_all.sh"
# SPECVIRT_EXIT_SCRIPT = "clean_all.sh"

# PRIME_HOST_INIT_SCRIPT[0] = "jAppInit.sh"
# PRIME_HOST_INIT_SCRIPT[1] = "Clean_webserver.sh"
# PRIME_HOST_INIT_SCRIPT[2] = "Clean_mailserver.sh"
# PRIME_HOST_INIT_SCRIPT[3] = "Clean_idleserver.sh"

# PRIME_HOST_EXIT_SCRIPT[0] = "japp_cleanup.sh"
# PRIME_HOST_EXIT_SCRIPT[1] = "web_cleanup.sh"
# PRIME_HOST_EXIT_SCRIPT[2] = "mail_cleanup.sh"
# PRIME_HOST_EXIT_SCRIPT[3] = "idle_cleanup.sh"

# The PRIME_HOST_RMI_PORT is the port through which the RMI calls are
sent to
# the prime client by the prime controller (specvirt). Note that if you
have
# more than one prime client on the same system, you MUST use different
port
# numbers for each (i.e. they do not share the same port)

PRIME_HOST_RMI_PORT[0][0] = 9900
PRIME_HOST_RMI_PORT[0][1] = 9901
PRIME_HOST_RMI_PORT[0][2] = 9902
PRIME_HOST_RMI_PORT[0][3] = 9903

PRIME_HOST_RMI_PORT[1][0] = 9910
PRIME_HOST_RMI_PORT[1][1] = 9911
PRIME_HOST_RMI_PORT[1][2] = 9912
PRIME_HOST_RMI_PORT[1][3] = 9913

PRIME_HOST_RMI_PORT[2][0] = 9920
PRIME_HOST_RMI_PORT[2][1] = 9921
PRIME_HOST_RMI_PORT[2][2] = 9922
PRIME_HOST_RMI_PORT[2][3] = 9923


# PRIME_PATH and CLIENT_PATH are the full paths to the prime client and
client,
# respectively, and the index corresponds to the workload index (i.e.
2nd index)
# used in the PRIME_HOST keys.

PRIME_PATH[0][0] = "C:/SPECvirt_sc2010/SPECjAppServer2004/classes"
PRIME_PATH[0][1] = "C:/SPECvirt_sc2010/SPECweb2005"
PRIME_PATH[0][2] = "C:/SPECvirt_sc2010/SPECimap"
PRIME_PATH[0][3] = "C:/SPECvirt_sc2010/SPECpoll"

PRIME_PATH[1][0] = "C:/SPECvirt_sc2010t1/SPECjAppServer2004/classes"
```

```
PRIME_PATH[1][1] = "C:/SPECvirt_sc2010t1/SPECweb2005"
PRIME_PATH[1][2] = "C:/SPECvirt_sc2010t1/SPECimap"
PRIME_PATH[1][3] = "C:/SPECvirt_sc2010t1/SPECpoll"

PRIME_PATH[2][0] = "C:/SPECvirt_sc2010t2/SPECjAppServer2004/classes"
PRIME_PATH[2][1] = "C:/SPECvirt_sc2010t2/SPECweb2005"
PRIME_PATH[2][2] = "C:/SPECvirt_sc2010t2/SPECimap"
PRIME_PATH[2][3] = "C:/SPECvirt_sc2010t2/SPECpoll"



POLL_PRIME_PATH = "C:/SPECvirt_sc2010/SPECpoll"

CLIENT_PATH[0][0] = "C:/SPECvirt_sc2010/SPECjAppServer2004/classes"
CLIENT_PATH[0][1] = "C:/SPECvirt_sc2010/SPECweb2005"
CLIENT_PATH[0][2] = "C:/SPECvirt_sc2010/SPECimap"
CLIENT_PATH[0][3] = "C:/SPECvirt_sc2010/SPECpoll"

CLIENT_PATH[1][0] = "C:/SPECvirt_sc2010t1/SPECjAppServer2004/classes"
CLIENT_PATH[1][1] = "C:/SPECvirt_sc2010t1/SPECweb2005"
CLIENT_PATH[1][2] = "C:/SPECvirt_sc2010t1/SPECimap"
CLIENT_PATH[1][3] = "C:/SPECvirt_sc2010t1/SPECpoll"

CLIENT_PATH[2][0] = "C:/SPECvirt_sc2010t2/SPECjAppServer2004/classes"
CLIENT_PATH[2][1] = "C:/SPECvirt_sc2010t2/SPECweb2005"
CLIENT_PATH[2][2] = "C:/SPECvirt_sc2010t2/SPECimap"
CLIENT_PATH[2][3] = "C:/SPECvirt_sc2010t2/SPECpoll"



POLL_CLIENT_PATH = "C:/SPECvirt_sc2010/SPECpoll"

# Use FILE_SEPARATOR if you want to override the use of the prime
client OS's
# file separator. (This may be required when using a product like
Cygwin on
# Windows.)
#
# FILE_SEPARATOR = "\"

# PRIME_APP is the prime client process that the clientmgr process will
start
# for each benchmark workload, with indexes corresponding to the
different
# workloads being run

PRIME_APP[0] = "org.spec.jappserver.launcher.jappserver"
PRIME_APP[1] = "-jar specweb.jar"
PRIME_APP[2] = "-jar specimap.jar -calibrate"
PRIME_APP[3] = "-jar specpoll.jar"

POLL_PRIME_APP = "-jar specpoll.jar"

# CLIENT_APP is the name of the client (driver) that is going to be
started
# by SPECprime and controlled by the prime client. Any arguments that
should
```

```
# be passed to the client application should follow the name. If you
want to
# specify different arguments for different tiles of the same workload,
use the
# CLIENT_APP[tile][wkload] format.
# Example: CLIENT_APP[0][1] = "-jar specwebclient.jar -lh webclient0"

CLIENT_APP[0][0] = "org.spec.jappserver.launcher.jappclient"
CLIENT_APP[0][1] = "-jar specwebclient.jar"
CLIENT_APP[0][2] = "-jar specimapclient.jar"
CLIENT_APP[0][3] = "-jar specpollclient.jar"

CLIENT_APP[1][0] = "org.spec.jappserver.launcher.jappclient"
CLIENT_APP[1][1] = "-jar specwebclient.jar"
CLIENT_APP[1][2] = "-jar specimapclient.jar"
CLIENT_APP[1][3] = "-jar specpollclient.jar"

CLIENT_APP[2][0] = "org.spec.jappserver.launcher.jappclient"
CLIENT_APP[2][1] = "-jar specwebclient.jar"
CLIENT_APP[2][2] = "-jar specimapclient.jar"
CLIENT_APP[2][3] = "-jar specpollclient.jar"



POLL_CLIENT_APP = "-jar specpollclient.jar"

# PRIME_START_DELAY is the number of seconds to wait after starting the
# clients before starting the prime clients. Increase this value if you
# find that prime clients fail to start because the clients have not
# finished preparing to listen for prime client commands before these
# commands are sent.

PRIME_START_DELAY = 20

# WORKLOAD_START_DELAY can be used to stagger the time at which clients
# begin to ramp up their client load by delaying client thread ramp-up
by
# the specified number of seconds. Seconds specified is *total* time
from
# the beginning of the client ramp-up phase. Therefore, if you have
# delays of 1, 5, and 3, repectively for three different clients,
# the order of the start of workload client ramp-up would be first,
# third, and then second.
# Format examples:
# Default format: all tiles/all workloads use this non-indexed delay
value
#                 unless otherwise specified.
# WORKLOAD_START_DELAY = 1
# Tile delay format: all workloads on Tile "x" use this value
# WORKLOAD_START_DELAY[x] = 1
# Tile/workload delay format: Workload "y" on Tile "x" uses this value
# WORKLOAD_START_DELAY[x][y] = 1

WORKLOAD_START_DELAY = 1
# WORKLOAD_START_DELAY[0] = 1
# WORKLOAD_START_DELAY[0][0] = 1
```

```
# RAMP_SECONDS and WARMUP_SECONDS supersede any values used in the
# workload-specific config files for ramp-up and warm-up time.
(RAMP_SECONDS
# overrides "triggerTime" in SPECjAppServer2004.) The minimum compliant
value
# for RAMPUP_SECONDS is 180. The minimum for WARMUP_SECONDS is 300.
# Format examples:
# Default format: all tiles/all workloads use this non-indexed delay
value
#                 unless otherwise specified.
# RAMP_SECONDS = 1
# WARMUP_SECONDS = 1
# Tile delay format: all workloads on Tile "x" use this value
# RAMP_SECONDS[x] = 1
# WARMUP_SECONDS[x] = 1
# Tile/workload delay format: Workload "y" on Tile "x" uses this value
# RAMP_SECONDS[x][y] = 1
# WARMUP_SECONDS[x][y] = 1

RAMP_SECONDS = 180
#RAMP_SECONDS[0] = 180
# RAMP_SECONDS[0][0] = 180

WARMUP_SECONDS= 300
# WARMUP_SECONDS[0] = 900
# WARMUP_SECONDS[0][0] = 900

# POLL_INTERVAL_SEC is the number of seconds that polling data should
be
# collected once polling starts. The minimum value for this property is
7200.

POLL_INTERVAL_SEC = 7200

# ECHO_POLL controls whether client polling values are mirrored on the
prime
# clients.

ECHO_POLL = 1

# DEBUG_LEVEL controls the amount of debug information displayed during
a
# benchmark run.

DEBUG_LEVEL = 10
# CLIENT_LISTENER_PORT is the port used by the clientmgr listener on
each
# physical client system (driver) to start the client processes for
each
# workload

CLIENT_LISTENER_PORT = "1088"

POLLING_RMI_PORT = "8001"

# The WORKLOAD_CLIENTS values are the client hostnames (or IP
addresses) and
```

```
# ports used by the workload clients. The hostname or IP address is
specified
# relative to the workload prime client, and not the SPECvirt
controller. For
# example, specifying 127.0.0.1 (or "localhost") tells the workload
prime
# client to run this client on *its* host OS's loopback interface,
rather than
# locally on the SPECvirt controller. If, for example, you use the
hostname
# "benchclient1" for all of your clients, and the corresponding prime
client
# resolves this name to unique IP addresses on each client used, then
these
# keys can be of the form WORKLOAD_CLIENTS[workload]. Otherwise, like
the
# PRIME_HOST keys, these need to be of the form
WORKLOAD_CLIENTS[tile][workload].
# Format examples:
# Workload-specific format: Workload "y" uses this value for all tiles
# WORKLOAD_CLIENTS[y] = "myhostname:1091"
# Tile/workload-specific format: Workload "y" on Tile "x" uses this
value
# WORKLOAD_CLIENTS[x][y] = "myhostname:1091"

WORKLOAD_CLIENTS[0][0] = "suganya1:1091"
WORKLOAD_CLIENTS[0][1] = "suganya1:1010"
WORKLOAD_CLIENTS[0][2] = "suganya1:1200"
WORKLOAD_CLIENTS[0][3] = "suganya1:1900"

WORKLOAD_CLIENTS[1][0] = "suganya2:2091"
WORKLOAD_CLIENTS[1][1] = "suganya2:2010"
WORKLOAD_CLIENTS[1][2] = "suganya2:2200"
WORKLOAD_CLIENTS[1][3] = "suganya2:2900"

WORKLOAD_CLIENTS[2][0] = "suganya3:3091"
WORKLOAD_CLIENTS[2][1] = "suganya3:3010"
WORKLOAD_CLIENTS[2][2] = "suganya3:3200"
WORKLOAD_CLIENTS[2][3] = "suganya3:3900"

# PRIME_CONFIG_FILE is the list of any files that need to be copied
from the
# corresponding LOCAL_CONFIG_DIR directory on the prime controller to
the
# PRIME_CONFIG_DIR directory on the corresponding PRIME_HOST.
# Valid format examples:
# Workload-specific format: file is copied to Workload "y" for all
tiles
# PRIME_CONFIG_FILE[y] = "myProps.config"
# Tile/workload-specific format: file is copied to Workload "y" on Tile
"x"
# PRIME_CONFIG_FILE[x][y] = "myProps.config"

PRIME_CONFIG_FILE[0][0] = "run.properties,glassfish.env"
PRIME_CONFIG_FILE[0][1] =
"Test.config,Testbed.config,SPECweb_Support.config"
```

```
PRIME_CONFIG_FILE[0][2] =
"IMAP_config.rc,IMAP_fixed.rc,IMAP_sysinfo.rc"
PRIME_CONFIG_FILE[0][3] = "Test.config"

PRIME_CONFIG_FILE[1][0] = "run.properties,glassfish.env"
PRIME_CONFIG_FILE[1][1] =
"Test.config,Testbed.config,SPECweb_Support.config"
PRIME_CONFIG_FILE[1][2] =
"IMAP_config.rc,IMAP_fixed.rc,IMAP_sysinfo.rc"
PRIME_CONFIG_FILE[1][3] = "Test.config"

PRIME_CONFIG_FILE[2][0] = "run.properties,glassfish.env"
PRIME_CONFIG_FILE[2][1] =
"Test.config,Testbed.config,SPECweb_Support.config"
PRIME_CONFIG_FILE[2][2] =
"IMAP_config.rc,IMAP_fixed.rc,IMAP_sysinfo.rc"
PRIME_CONFIG_FILE[2][3] = "Test.config"

POLL_CONFIG_FILE = "Test.config"

# LOCAL_CONFIG_DIR is the *source* location for the configuration files
to be
# copied to the prime clients. PRIME_CONFIG_DIR is the *target*
location for
# the config files copied from the source location.
# Valid format examples:
# Workload-specific format: directory path is used for Workload "y" for
all tiles
# LOCAL_CONFIG_DIR[y] = "/my/source/path"
# PRIME_CONFIG_DIR[y] = "/my/target/path"
# Tile/workload-specific format: directory is used for Workload "y" on
Tile "x"
# LOCAL_CONFIG_DIR[x][y] = "/my/source/path"
# PRIME_CONFIG_DIR[x][y] = "/my/target/path"

LOCAL_CONFIG_DIR[0][0] = "C:/SPECvirt_sc2010/SPECjAppServer2004/config"
LOCAL_CONFIG_DIR[0][1] = "C:/SPECvirt_sc2010/SPECweb2005"
LOCAL_CONFIG_DIR[0][2] = "C:/SPECvirt_sc2010/SPECimap"
LOCAL_CONFIG_DIR[0][3] = "C:/SPECvirt_sc2010/SPECpoll"

LOCAL_CONFIG_DIR[1][0] =
"C:/SPECvirt_sc2010t1/SPECjAppServer2004/config"
LOCAL_CONFIG_DIR[1][1] = "C:/SPECvirt_sc2010t1/SPECweb2005"
LOCAL_CONFIG_DIR[1][2] = "C:/SPECvirt_sc2010t1/SPECimap"
LOCAL_CONFIG_DIR[1][3] = "C:/SPECvirt_sc2010t1/SPECpoll"

LOCAL_CONFIG_DIR[2][0] =
"C:/SPECvirt_sc2010t2/SPECjAppServer2004/config"
LOCAL_CONFIG_DIR[2][1] = "C:/SPECvirt_sc2010t2/SPECweb2005"
LOCAL_CONFIG_DIR[2][2] = "C:/SPECvirt_sc2010t2/SPECimap"
LOCAL_CONFIG_DIR[2][3] = "C:/SPECvirt_sc2010t2/SPECpoll"

POLL_LOCAL_CFG_DIR = "C:/SPECvirt_sc2010/SPECpoll"

PRIME_CONFIG_DIR[0][0] = "C:/SPECvirt_sc2010/SPECjAppServer2004/config"
PRIME_CONFIG_DIR[0][1] = "C:/SPECvirt_sc2010/SPECweb2005"
PRIME_CONFIG_DIR[0][2] = "C:/SPECvirt_sc2010/SPECimap"
```

```
PRIME_CONFIG_DIR[0][3] = "C:/SPECvirt_sc2010/SPECpoll"

PRIME_CONFIG_DIR[1][0] =
"C:/SPECvirt_sc2010t1/SPECjAppServer2004/config"
PRIME_CONFIG_DIR[1][1] = "C:/SPECvirt_sc2010t1/SPECweb2005"
PRIME_CONFIG_DIR[1][2] = "C:/SPECvirt_sc2010t1/SPECimap"
PRIME_CONFIG_DIR[1][3] = "C:/SPECvirt_sc2010t1/SPECpoll"

PRIME_CONFIG_DIR[2][0] =
"C:/SPECvirt_sc2010t2/SPECjAppServer2004/config"
PRIME_CONFIG_DIR[2][1] = "C:/SPECvirt_sc2010t2/SPECweb2005"
PRIME_CONFIG_DIR[2][2] = "C:/SPECvirt_sc2010t2/SPECimap"
PRIME_CONFIG_DIR[2][3] = "C:/SPECvirt_sc2010t2/SPECpoll"

POLL_PRIME_CFG_DIR = "C:/SPECvirt_sc2010/SPECpoll"

# Setting USE_RESULT_SUBDIRS to 1 puts each result set in a different
results
# subdirectory with a unique timestamp-based name. Setting to 0 will
not create
# a unique subdirectory, and any earlier results in the parent
"results"
# directory will be overwritten by newer test results. Accordingly,
setting
# USE_RESULT_SUBDIRS to 0 is only recommended for use with Faban. And
# conversely, setting USE_RESULT_SUBDIRS to 1 is *not* recommended when
using
# Faban.

USE_RESULT_SUBDIRS = 1

# USE_PTDS controls whether or not the power/temp daemons (PTDs) are
used during
# the benchmark. Set to 0 to run without taking power or temperature
# measurements. PTD_HOST is the hostname of the system running the PTD.
For
# more than one PTD, copy, paste, and increment the index for each PTD.
# PTD_PORT is the corresponding port the PTD is listening on.
PTD_TARGET is
# the type of component the power/temp meter is monitoring. ("SUT"
identifies
# meter as monitoring a main system/server; "EXT_STOR" identifies meter
as
# monitoring any external storage.) Setting SAMPLE_RATE_OVERRIDE for
any PTD
# allows you to override the default sample rate for the power or
temperature
# meter. This is NOT allowed for compliant runs. However, if
overridden,
# OVERRIDE_RATE_MS is the sample rate used instead of the meter's
default, in
# milliseconds. LOCAL_HOSTNAME and LOCAL_PORT are used to specify the
*local*
# network interface and port to use to connect with the PTD_HOST. In
most cases,
# specifying these values is neither necessary nor recommended. So
leave them
```

```
# commented out unless necessary.

USE_PTDS = 0

# PTD_HOST[0] = myPtdHostname-0
# PTD_PORT[0] = 8888
# PTD_TARGET[0] = "SUT"
# LOCAL_HOSTNAME[0] = localInterface-0
# LOCAL_PORT[0] = 0
# SAMPLE_RATE_OVERRIDE is a "fixed" workload property. It is included
here to
# keep properties related to the same index in the same place.
# SAMPLE_RATE_OVERRIDE[0] = "0"
# OVERRIDE_RATE_MS[0] = 1000

# PTD_HOST[1] = myPtdHostname-1
# PTD_PORT[1] = 8889
# PTD_TARGET[1] = "EXT_STOR"
# LOCAL_HOSTNAME[1] = localInterface-1
# LOCAL_PORT[1] = 0
# SAMPLE_RATE_OVERRIDE is a "fixed" workload property. It is included
here to
# keep properties related to the same index in the same place.
# SAMPLE_RATE_OVERRIDE[1] = "0"
# OVERRIDE_RATE_MS[1] = 1000

# PTD_HOST[2] = myPtdHostname-2
# PTD_PORT[2] = 8890
# PTD_TARGET[2] = "SUT"
# LOCAL_HOSTNAME[2] = "localInterface-2"
# LOCAL_PORT[2] = 0
# SAMPLE_RATE_OVERRIDE is a "fixed" workload property. It is included
here to
# keep properties related to the same index in the same place.
# SAMPLE_RATE_OVERRIDE[2] = "0"
# OVERRIDE_RATE_MS[2] = 1000

# Use RESULT_TYPE to indicate the type of result submission (or
combination of
# submissions) that you would like to create.
#
# value: the value used for RESULT_TYPE
# perf : generate a non-power report (with SPECvirt_sc2009 metric)
# ppw  : generate a SUT power-performance report (with
SPECvirt_sc2009_PPW metric)
# ppws : generate a server-only (primary metric includes server power
only)
#        power-performance report (with SPECvirt_sc2009_ServerPPW
metric)
#
# Possible values are:
#
# value | perf | ppw | ppws
#   1   |  x   |     |
#   2   |      |  x  |
#   3   |  x   |  x  |
#   4   |      |     |  x
```

```
#   5   |  x   |      |  x
#   6   |      |  x   |  x
#   7   |  x   |  x   |  x
#
RESULT_TYPE = 1

# Tile-specific format of the fixed property, LOAD_SCALE_FACTORS. Tile
"x" runs
# at the specified load scaling factor. Compliant values are between
0.1 and
# 0.9 in increments of 0.1. This property is intended to allow for one
tile to
# run at reduced load. Defining more than one tile to run at a reduced
load,
# or for any tile to run at greater-than-full'load (i.e.
LOAD_SCALE_FACTORS
# value > 1.0) will result in a non-compliant run.

#LOAD_SCALE_FACTORS[0] = "0.9"
#LOAD_SCALE_FACTORS[1] = "1.0"



# Setting IGNORE_CLOCK_SKEW to "1" causes the prime controller to skip
the
# system clock synchronization check at the beginning of a benchmark
run.
# Setting to "0" (default) means the prime controller and the prime
clients
# perform this check to assure all prime clients, clients, and VMs are
in time
# sync with the prime controller. If set to 1, CLOCK_SKEW_ALLOWED is
the number
# of seconds of clock skew the prime controller and prime clients will
allow at
# the beginning of a benchmark run without aborting.

IGNORE_CLOCK_SKEW = 0
CLOCK_SKEW_ALLOWED = 5


##########################################################################
#
# FIXED WORKLOAD PROPERTIES
# Changing any of the property values, below, will result in a
# non-compliant benchmark run
##########################################################################
#

# Virtual machines (VMs) are added in units called tiles. VMS_PER_TILE
is the
# number of VMs contained in a single tile.

VMS_PER_TILE = 6

NUM_WORKLOADS =4
```

```
WORKLOAD_LABEL[0] = "Application Server"
WORKLOAD_LABEL[1] = "Web Server"
WORKLOAD_LABEL[2] =  "Mail Server"
WORKLOAD_LABEL[3] = "Idle Server"

BACKEND_VM_LABEL[0] = "Database Server"
BACKEND_VM_LABEL[1] = "Infrastructure Server"

# POLL_MASTERS controls whether or not to request polling data from the
prime
# clients.

POLL_MASTERS = 1

# IDLE_RAMP_SEC, IDLE_WARMUP_SEC, and IDLE_POLL_SEC are the ramp,
warmup, and
# polling/runtime values used for the active-idle measurement phase
only.
# IDLE_START_DELAY is the active idle phase equivalent of
WORKLOAD_START_DELAY.

IDLE_START_DELAY = 0
IDLE_RAMP_SEC = 10
IDLE_WARMUP_SEC = 10
IDLE_POLL_SEC = 600

# Set INTERVAL_POLL_VALUES = 0 for cumulative polling data over the
entire
# measurement interval. Set it to 1 if you want only the polling data
that
# is added between polling intervals. (Note: some workloads do not
support
# polling-interval-based results reporting and will ignore a non-zero
value.
# Default value: 0.

INTERVAL_POLL_VALUES = 0

# POLL_DELAY_SEC is the number of seconds after *all* prime clients
have
# started running that the prime controller should wait before starting
to
# request polling data.

POLL_DELAY_SEC = 10

# BEAT_INTERVAL is the number of seconds between prime client pollings.
# This controls either the frequency that prime client data is returned
to
# the prime controller (if POLL_MASTERS is set to 1). This value must
not
# be less than the greatest value used by the prime clients for their
runs.

BEAT_INTERVAL = 10
```

```
# If clients are not returning polling data as required, the prime
controller
# will abort the run. Set IGNORE_POLL_ERROR to 1 if you want to prevent
the
# run from aborting.

IGNORE_POLL_ERROR = 0

# RESULT_FILE_NAMES are the names of the results files created by the
workload
# that the prime controller should collect from the prime clients after
a run
# has completed. The indexes correspond with the workload indexes.

RESULT_FILE_NAMES[0] = "Atomicity.html, Audit.report, Dealer.detail,
Dealer.summary, Mfg.detail, Mfg.summary, result.props,
SPECjAppServer.summary"
RESULT_FILE_NAMES[1] = "SPECweb_Support.raw"
RESULT_FILE_NAMES[2] = "output.raw, specimap.rsl"
RESULT_FILE_NAMES[3] = "SPECpoll.raw"

POLL_RES_FILE_NAMES = "SPECpoll.raw"

# USE_WEIGHTED_QOS controls the manner of calculating QOS for the
workloads. A
# value of 0 means to apply the same weight to all QOS-related fields
used to
# calculate the aggregate QOS value. A value of 1 (or higher) results
in a
# weighted QOS based on frequency being used to calculate aggregate
QOS.

USE_WEIGHTED_QOS = 0

# Set PTD_POLL to 1 in order to poll the PTDs during the POLL_INTERVAL.
Note
# that this property has no effect when USE_PTDS is set to 0, so it
should be
# left at the value 1 even for performance-only benchmark runs.
POWER_POLL_VAL
# selects which value to poll from the power meter (possible values:
"Watts",
# "Volts", "Amps", "PF"). Similarly, TEMP_POLL_VAL controls which value
to poll
# from the temperature meter (options: "Temperature", "Humidity").

PTD_POLL = 1
POWER_POLL_VAL = "Watts"
TEMP_POLL_VAL = "Temperature"

# LOAD_SCALE_FACTORS is the list of multipliers to the load levels for
the
# individual workload levels. For each value and in the order listed,
the
# benchmark harness will run a full run at the calculated load rate
with a
# QUIESCE_SECONDS wait interval between each point.
```

```
# LOAD_SCALE_FACTORS format examples:
# Default format: all tiles run with this (set of) load scaling factors
# LOAD_SCALE_FACTORS = "1.0,0"

LOAD_SCALE_FACTORS = "1.0,0"
QUIESCE_SECONDS = 300

# WORKLOAD_SCORE_TMAX_VALUE is the theoretical maximum throughput rate
for each
# workload. Comment these values out if you do not want to normalize
scores to
# the theoretical max. Setting the value to 0 has the effect of not
using this
# workload's score in calculating the result.

WORKLOAD_SCORE_TMAX_VALUE[0] = 34.86
WORKLOAD_SCORE_TMAX_VALUE[1] = 54.17
WORKLOAD_SCORE_TMAX_VALUE[2] = 89.93
WORKLOAD_SCORE_TMAX_VALUE[3] = 0

# WORKLOAD_LOAD_LEVEL supercedes any values used in the workload-
specific
# configuration files to control client load. For the jApp workload,
txRate is
# overwritten with this value. For web, SIMULTANEOUS_SESSIONS is
overwritten.
# For imap, the number of users is set to this value.

WORKLOAD_LOAD_LEVEL[0] = 20
WORKLOAD_LOAD_LEVEL[1] = 500
WORKLOAD_LOAD_LEVEL[2] = 500
WORKLOAD_LOAD_LEVEL[3] = 0
```

VITA

Suganya Sridharan has a Bachelor of Engineering from Thiagarajar college of Engineering affiliated to Anna University, India in Computer Science and Engineering, 2007 and expects to receive a Master of Science in Computer and Information Sciences from the University of North Florida, August 2012. Dr. Sanjay Ahuja of the University of North Florida is Suganya‘s thesis advisor. Suganya has been a Technical Support Engineer specializing in Java based product Response at KANA Software Inc., Sunnyvale, California for the past ten months. Suganya has 2+ years experience at Computer Sciences Corporation (CSC) India using ASP.Net (C#), SQL Server 2008 and Microsoft Reporting Services. She also has internship experience of about 7 months at Blue Cross Blue Shield of Florida, Jacksonville, Florida, working for automation testing using QTP. Additionally, she was an intern for over two months at Synopsys Inc., Mountain View, California, working on Microsoft Reporting services. Suganya was a Graduate Teaching Assistant at University of North Florida, School of Computing, in 2010-2011. She has the following certifications 1) IBM Certified Database Associate DB2 Universal Database v8.1 Family Fundamentals 2) IBM Certified Associate Developer Web Sphere StudioV5.0. Suganya‘s academic work has included use of Java, C, C++, Microsoft Visual Studio.Net, and SQL.

Suganya likes to incorporate her computer skills in real life situations needing the use of Data Mining and Cloud Computing. Her main goal is to become a respected IT professional who strives to produce excellence in every area useful for the society, and to maintain ethical and respectful approaches to working with the vast power of computing.