

Master Thesis
Electrical Engineering
December 2013



**Performance comparison of KVM and XEN for
telecommunication services**

Siavash Outadi
Jana Trchalikova

School of Computing
Blekinge Institute of Technology
37179 Karlskrona
Sweden

This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Electrical Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information

Authors:

Siavash Outadi

E-mail: siavash.outadi@gmail.com

Jana Trchalikova

E-mail: jana.trchalikova@gmail.com

External Adviser:

Stefan Wallenberg

IT Project Manager

DCI/DAC/D

Ericsson AB

371 23 KARLSKRONA SWEDEN

Internet: www.ericsson.com

Phone: +46 455 395000

University Adviser:

Dr. David Erman

Senior Lecturer

School of Computing

Blekinge Institute of Technology

371 79 KARLSKRONA SWEDEN

Internet: www.bth.se/com

Phone: +46 455 385000

Abstract

High stability of telecommunication services has a positive effect on customer satisfaction and thus helps to maintain competitiveness of the product in telecommunication market. Since live migration provides a minimal downtime of virtual machines, it is deployed by telecommunication companies to ensure high availability of services and to prevent service interruptions.

The main objective of this research is to assess the performance of various hypervisors in terms of live migration and determine which of them best meets the criteria given by a telecommunication company. Response time and CPU utilization of telecommunication services are measured in non-virtualized and virtualized environments to better understand the impacts of virtualization on the services. Two hypervisors, i.e. KVM and XEN, are used to grasp their characteristic behaviour of handling the services. Furthermore, performance of live migration is assessed for both hypervisors using miscellaneous test cases to identify which one has the best overall performance in terms of downtime and total migration time.

Keywords: Cloud computing, virtualization, telecommunications, live migration, hypervisor.

Acknowledgements

First of all, we would like to express grateful thanks to our supervisor David Erman who used his comprehensive knowledge and academic experience to support us throughout our studies. We would also like to thank him for his patience and careful guidance of our research team during preparation of the thesis.

It is also great pleasure for us to thank Lars Lundberg and Dragos Ilie for their in-depth comments and valuable advice regarding benchmark definition and research methodology.

A special thanks goes to our external advisors Stefan Wallenberg, Joakim Jaxér, Jim Håkansson, Martin Westermark and Peter Ketzenius for technical and managerial support. Our external advisors have invested a large amount of their effort in guiding us to achieve the thesis objectives.

Last but not least, we would like to express our deepest gratitude to our families for their love and support they gave us throughout our studies.

Contents

Abstract	i
Acknowledgements	ii
Contents	iii
List of Figures	vi
List of Tables	vii
List of Abbreviations	viii
Introduction	1
1 Introduction	1
1.1 Cloud computing overview	1
1.2 Problem Identification and Motivation	2
1.3 Research questions	3
1.4 Thesis Outline	3
Background	3
2 Background	4
2.1 Literature Review	4
2.2 Cloud computing	5
2.3 Virtualization	6
2.4 Migration	7
2.5 Hypervisor	9

2.6	Signalling System No. 7	10
2.7	SIGTRAN	10
Experimental setup		11
3	Experimental Setup	12
3.1	Testbed	12
3.2	Cluster application	12
3.3	Simulators	13
3.4	Measurement tools	13
3.5	Non-virtualized environment setup	13
3.6	Virtualized environment setup	13
3.6.1	KVM	14
3.6.2	XEN	14
3.7	Test cases	15
3.7.1	Measurement procedure	15
3.7.2	Test cases in non-virtualized scenario	16
3.7.3	Test cases in virtualized scenario	17
3.7.4	Migration test cases	19
Results		20
4	Results	21
4.1	Virtual vs. non-virtual scenario	21
4.1.1	KVM	21
4.1.2	XEN	23
4.2	Migration scenario	24
4.2.1	KVM	24
4.2.2	XEN	26
4.3	Result summary	27
4.3.1	Non-virtualized vs virtualized environments	28
4.3.2	Various CPU core setups	28
4.3.3	Live migration	29
4.4	Statistical analysis	29
4.4.1	Formulas	29
4.4.2	Non-virtualized environment	31
4.4.3	KVM	31
4.4.4	XEN	33

Conclusions	34
5 Conclusions	35
5.1 Future work	36
Appendix	36
A Commands	37
A.1 CPU utilization	37
A.1.1 Non-virtualized environment	37
A.1.2 KVM	37
A.1.3 XEN	37
A.2 Live migration	37
A.2.1 KVM	37
A.2.2 XEN	37
B MATLAB code samples	38
B.1 CPU utilization	38
B.1.1 Non-virtualized environment	38
B.1.2 KVM, the 16 core setup	39
B.1.3 XEN, the 16 core setup	41
B.2 Response time	43
B.2.1 Non-virtualized environment	43
B.2.2 KVM, the 16 core setup	44
B.2.3 XEN, the 16 core setup	46

List of Figures

2.1	Logical steps of live migration	8
2.2	The two types of hypervisors	9
2.3	SIGTRAN protocol architecture	11
3.1	Non-virtualized testbed	14
3.2	Virtualized testbed	15
3.3	Migration test case	20
4.1	KVM - CPU utilization	22
4.2	KVM - Average response time	22
4.3	XEN - CPU utilization	23
4.4	XEN - Average response time	24
4.5	KVM - CPU utilization during the migration process	25
4.6	KVM - Maximum response time during the migration process	25
4.7	XEN - CPU utilization during the migration process	26
4.8	XEN - Maximum response time during the migration process	27
B.1	A sample from CPU data file for non-virtualized environment	38
B.2	A sample from CPU data file for KVM	39
B.3	A sample from CPU data file for XEN	41
B.4	A sample from response time data file for non-virtualized environment	43
B.5	A sample from response time data file for KVM	45
B.6	A sample from response time data file for XEN	46

List of Tables

3.1	Test cases in the non-virtualized scenario	17
3.2	KVM - Test cases for the 16 core setup	17
3.3	KVM - Test cases for the 12 core setup	18
3.4	KVM - Test cases for the 6 core setup	18
3.5	XEN - Test cases for the 16 core setup	18
3.6	XEN - Test cases for the 12 core setup	19
3.7	XEN - Test cases for the 6 core setup	19
4.1	Non-virtualized vs virtualized environment - CPU utilization	28
4.2	Non-virtualized vs virtualized environment - response time . .	28
4.3	KVM - CPU utilization overview	28
4.4	XEN - CPU utilization overview	28
4.5	KVM - Response time overview	29
4.6	XEN - Response time overview	29
4.7	Live migration measurements	29
4.8	CPU data analysis for the non-virtualized environment	31
4.9	Response time analysis for the non-virtualized environment . .	31
4.10	KVM - CPU data analysis for the 16 core setup	31
4.11	KVM - CPU data analysis for the 12 core setup	32
4.12	KVM - CPU data analysis for the 6 core setup	32
4.13	KVM - Response time data analysis for the 16 core setup . .	32
4.14	KVM - Response time data analysis for the 12 core setup . .	33
4.15	KVM - Response time data analysis for the 6 core setup . . .	33
4.16	XEN - CPU data analysis for the 16 core setup	33
4.17	XEN - CPU data analysis for the 12 core setup	33
4.18	XEN - CPU data analysis for the 6 core setup	34
4.19	XEN - Response time data analysis for the 16 core setup . . .	34
4.20	XEN - Response time data analysis for the 12 core setup . . .	34
4.21	XEN - Response time data analysis for the 6 core setup . . .	34

List of Abbreviations

<i>CPU</i>	Central processing unit
<i>GB</i>	Gigabyte
<i>GFS2</i>	Global File System 2
<i>GUI</i>	Graphical user interface
<i>HPC</i>	High Performance Computing
<i>I/O</i>	Input/output
<i>IT</i>	Information technology
<i>KVM</i>	Kernel-based Virtual Machine, an open source hypervisor
<i>LUN</i>	Logical unit number
<i>OCFS2</i>	Oracle Cluster File System 2
<i>OS</i>	Operating system
<i>PVHVM</i>	para-virtualized on hardware virtualized machine
<i>QoS</i>	Quality of service
<i>RAM</i>	Random access memory
<i>RTO</i>	retransmission timeout
<i>SLA</i>	Service Level Agreement
<i>SLES</i>	SUSE Linux Enterprise Server
<i>SS7</i>	Signalling System No. 7 protocol
<i>TCP</i>	transmission control protocol
<i>VM</i>	Virtual machine
<i>VMM</i>	Virtual machine monitor

Chapter 1

Introduction

1.1 Cloud computing overview

Cloud computing is the next computing model after the grid, utility and distributed computing and it is getting more popular day by day. Nowadays, users have to cope with miscellaneous configurations and installations caused by complex IT infrastructures. Hence providing computing as a service, not as a product, is an effective solution for users to manage complex IT infrastructures [1]. The main idea of cloud computing is to deliver shared resources, software and information to devices as a utility over a network. It has many benefits such as scalability, good performance, high availability, more mobility and last but not least low cost. This causes a growing tendency in usage of cloud computing in the industry.

Virtualization as one of the main constituents of cloud computing provides many advantages in sharing and management of resources in the cloud [2]. For instance, it provides consolidation of clustered hardware into a single coherent management domain which remarkably improves manageability [5]. In a simplified way, virtualization is a technique to abstract the hardware and system resources from a given OS [3]. This is done by adding a virtualization layer which brings more flexibility and improved performance and enables installation of isolated containers called virtual machines (VMs). Isolation of the resources in the cloud computing brings many benefits, primarily higher hardware utilization and easier hardware management [3]. Multiple VMs can run on a single piece of hardware, increasing the utilization of the hardware while reducing the total number of required hardware units. Furthermore, virtualization technology makes it simple to migrate VMs across physical machines [4].

This Master Thesis is focused on live migration of VMs. Live migration is a capability to move a VM and its corresponding applications and data between two physical boxes with almost no service interruption. Therefore it considerably affects the QoS. Besides, live migration allows separation of

hardware and software handling which makes it a powerful tool for cluster administrators [5]. Migration also reduces overload of servers and facilitates their maintenance.

1.2 Problem Identification and Motivation

Telecommunications is one of the industry branches where high stability of services is very important. Nowadays, telecommunication industry must cope with cost and performance pressure and above all with demand for rapid development and deployment of new services. To keep up with new requirements and expectations, fundamental changes are required in telecommunication architecture.

As it has been mentioned before, cloud computing provides numerous advantages and therefore embracing cloud computing solutions is essential to remain competitive in the telecommunication market [8]. Therefore reaping the profits of cloud computing for IT optimization is one of the priorities of current research activities in telecommunication industry [7].

The level of services and delivery expectations are defined in Service Level Agreement (SLA). For instance, " policies such as 99,999% availability permits only 5 minutes of downtime a year " [9]. Thus a short downtime plays a key role in evaluation of telecommunication services in terms of QoS. Live migration ensures good QoS by reducing the downtime. The possibility to migrate VMs without a perceivable downtime is becoming more and more crucial in order to provide stable services [6].

This thesis is relevant to telecommunication companies that work with mobile networks and telecommunication systems. The main objective of this research is to assess the performance of various hypervisors in terms of live migration and determine which of them best meets the criteria given by a telecommunication company such as minimum downtime, response time and CPU utilization. Until now, no study of live migration performance focusing on telecommunication industry is available in open literature.

1.3 Research questions

Four research questions were formulated:

1. How does virtualization impact the response time and CPU utilization of telecommunication services?
2. Which hypervisor has better performance in terms of migration, response time and CPU utilization?
3. Are there any significant differences in performance of the hypervisors?
4. How do performances of the hypervisors differ for various test cases defined in section 3.7 of Chapter 3?

1.4 Thesis Outline

The thesis is outlined as follows. Chapter two covers related research work and background information regarding cloud computing, virtualization, migration and hypervisors. In chapter three, experimental setup is discussed in details. Results are reported and analysed in chapter four and conclusions are presented in fifth chapter.

Chapter 2

Background

2.1 Literature Review

Performance comparison of VMMs has been extensively studied in [10], [15], [20].

Jianhua Che et al. performed a quantitative and qualitative performance comparison of XEN and KVM using LINPACK, LM-bench and IOzone to identify the virtualization overhead [20].

The macro-performance and micro-performance of three VMMs (OpenVZ, XEN and KVM) are measured and analysed to distinguish their impact on virtualization systems. Virtualization of processor, memory, disk, network and server applications is categorized as macro-performance while virtualization of system operation and context switch is classified into micro-performance [10].

Performances of XEN, VMWare, KVM and VirtualBox are compared as well as features which they provide. The applicability of HPC Cloud environments to use FutureGrid resources is the main focus of the performance analysis [15].

The effects of live migration on the performance of the Multi-tier web application has been studied in detail in [11], [26].

Olio is used as a Web 2.0 application and Faban load generator represents an application and workload set. The metrics are chosen according to Service Level Agreement defined in Cloudstone [26].

Evaluation functions are proposed to estimate the impacts of live migration on multi-tier application with considering downtime of VMs and TCP's RTO as parameters. Besides, an experimental setup is conducted which separates the network for user request traffic from the network for live migration [11].

High performance OS migration is designed, implemented and evaluated for XEN with focus on data centre and cluster environments in [5]. Issues and

trade-offs involved in live local-area migration such as downtime and total migration time are addressed in the design and implementation. Pre-copy approach is selected for the migration and page level protection hardware is employed to ensure that a consistent snapshot is transferred. The impact of migration traffic on running services is controlled by using rate-adaptive algorithm. Evaluation is performed for several services. SPECweb99, a complex application-level benchmark, is used to produce a heavy Apache workload. Moreover, Quake 3 which is a multi-player on-line game server is used by six players for the experiment. Additionally, another experiment is conducted using MMuncher to produce a situation in which writing to memory by a virtual machine is faster than network transmission [5].

Performance models of live migration are introduced to predict the migration costs [9], [12], [13].

Two application-oblivious models are constructed to predict the migration cost in terms of performance and energy. The prediction is based on the gained knowledge about workloads at hypervisor level. It follows a quantitative approach. The evaluation of the models is accomplished using five representative workloads on XEN [12].

A performance model is designed using PRISM, a probabilistic model checker, based on the data gathered from the experiment regarding performance characteristics of live migrations. Furthermore, stack in sender server and heavy load in receiver server properties are considered for quantitative verification of the models [13].

Migration times are examined in detail and two simulation models are proposed and validated by measurements. These models can be used to predict service interruptions based on a specific workload [9].

2.2 Cloud computing

Cloud computing is a service oriented model and abstraction and accessibility are two crucial factors in this model. The underlying cloud architecture is abstracted and hidden from the user as a result of virtualization and consolidation. Concurrently, the key components of underlying cloud architecture can be easily accessed. In general, cloud computing delivers following services in a transparent way [15], [16]:

- Software-as-a-service
- Hardware-as-a-service
- Data-as-a-service
- Platform-as-a-service

There are four types of clouds [17], [18]:

1. *Private cloud*: Private cloud service model is operated solely for a specific organisation. It can be hosted internally or externally. This type of cloud service is more expensive since it requires more involvement for the organisation to virtualize the business environment.
2. *Public cloud*: Public cloud is a cloud which is made available to the general public. Its customers share the same infrastructure pool which makes this type of cloud very vulnerable and insecure. Public clouds operate on a low-cost or pay-per-use model. Public clouds can be accessed only via Internet.
3. *Community cloud*: Community cloud service model is shared between several organisations within the same community. Its main purpose is to bring benefits of public cloud with added level of security of private cloud. Community clouds can be either on-premise (local) or off-premise (remote).
4. *Hybrid cloud*: Hybrid cloud is a composition of two or more clouds (private, community or public) that are handled as unique entities but are bound together [19].

2.3 Virtualization

Virtualization refers to "abstraction of logical resources away from their underlying physical resources" [19]. Virtualization plays a significant role in cloud computing since it provides many advantages in sharing, management and isolation of the resources in the cloud [2]. Some of the important benefits of using virtualization are discussed below [3], [20], [21]:

- *Flexibility and Scalability*: Cloud computing follows the pay-per-use model which means the users will pay for the resources that they really use. However, the possibility of a rise of potential resources for future demands remains permissible due to the virtualization. Therefore it is very flexible and comes with a great agility and scalability.
- *Cost Effectiveness*: Virtualization reduces hardware requirements by employing resource consolidation, primarily server consolidation. Additionally, a dynamic provisioning is another capability which is feasible due to utilization of virtualized resources. As mentioned above, virtualization is a very cost effective solution for clustered environments.
- *Isolation*: It provides resource isolation including OS level and application level isolation.

The three main types of virtualization are as follows [14], [22], [23]:

1. *Full virtualization*: Full virtualization is based on emulation of the hardware. In this approach, the guest operating systems do not require any modification since they are not aware of being virtualized. It provides security and isolation for virtual machines and also facilitates migration and portability.
2. *Paravirtualization*: In paravirtualization, the guest OSes are aware of the hypervisor and the OS kernel is modified to provide an interface for communication between the hypervisor and the OS kernel which improves performance and efficiency. Compatibility and portability of para-virtualization is poor since it does not support unmodified OSes.
3. *Hardware assisted virtualization*: Hardware assisted virtualization uses virtualization hardware extensions, mainly host CPU, to provide full virtualization. Consequently, it needs an explicit support in host CPU which is not available in all processors. Intel VT and AMD-V processors include virtualization technology support.

Guests that are using this technique are usually slower than the guests which are using para-virtualization due to a high CPU overhead caused by emulation. However, hardware assisted virtualization does not require modification of the guest OS.

2.4 Migration

Migration is an important capability of virtualization. Migration process enables transferring memory images from the overloaded server to the destination server [6].

VM migration can be performed either as cold or live migration.

- *Cold migration* involves pausing a VM, copying all its memory to the target host and then restarting the VM on the target host. Cold migration has a very low complexity, but it can cause a considerable service downtime [24].
- During *live migration*, memory of a running VM is being repeatedly pre-copied until the difference between memory states is minimal. Afterwards, both VM are terminated, the memory residue is copied and the VM is restarted on the target host [25]. The main advantage of live migration is its small downtime.

As it has been mentioned, the process of copying of the memory in live migration is accomplished while the OS continues running which causes a less downtime. Consequently, it improves the QoS and performance of

the services and applications. Manageability, fault tolerance and improved performance are some other advantages of live migration [26]. The logical steps of live migration according to [5] is depicted in figure 2.1.

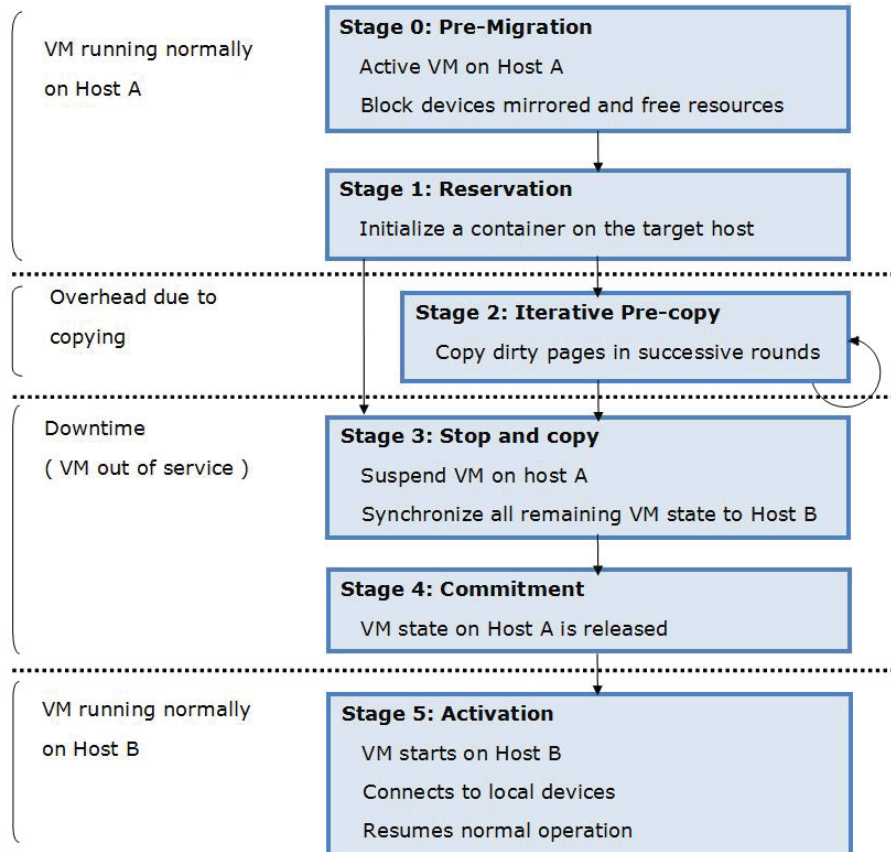


Figure 2.1: Logical steps of live migration

2.5 Hypervisor

A hypervisor, also known as a virtual machine manager, is one of the hardware virtualization techniques enabling multiple operating systems to share a single hardware platform [27]. Each operating system managed by the hypervisor is called a guest operating system. The main purpose of a hypervisor is to allocate host system resources according to requirements of guest operating systems. A variety of operating systems can be installed on one hypervisor in parallel, i.e. Windows, Linux, NetBSD.

There are two types of hypervisors:

- *Type 1* (native or bare metal) is a hypervisor which runs directly on the host hardware. The hypervisor has a direct access to all host system resources. Therefore hypervisors of type 1 offer a higher level of virtualization efficiency and security [28].
- *Type 2* (hosted) is a hypervisor which runs on a host operating system that provides virtualization services [29].

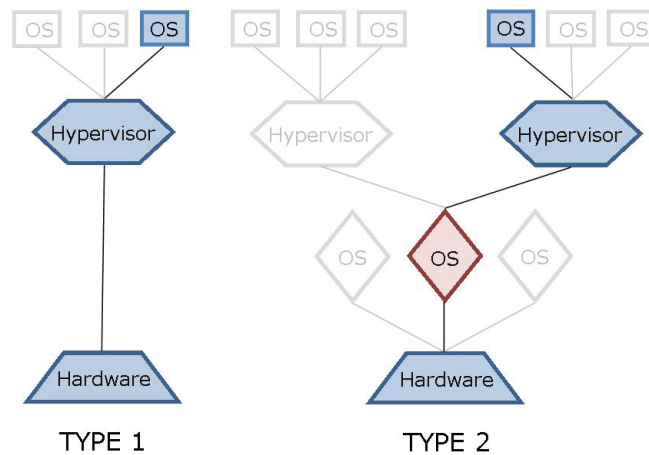


Figure 2.2: The two types of hypervisors

Different hypervisors are described in details below:

1. *KVM*: KVM (Kernel-based Virtual Machine) is an open source type 2 hypervisor which provides full virtualization for x86 hardware with virtualization extension (Hardware assisted) [29]. The core virtualization infrastructure is supplied by a loadable kernel module which is included in the releases since version 2.6.20.
2. *XEN*: An open source virtual machine monitor originally developed at the University of Cambridge. It is a type 1 hypervisor that runs directly on the hardware and is responsible for handling CPU and memory. It supports full virtualization using hardware assisted virtualization and also para-virtualization [30]. Additionally, there is a possibility to use para-virtualization techniques on the guests using hardware assisted virtualization to get an optimal performance. Using this approach, the disk emulation and network I/O are bypassed which causes the better performance.

2.6 Signalling System No. 7

Signalling System no. 7 (SS7) consists of several telephony signalling protocols used for control, management and maintenance signalling of telecommunication services [31], [32]. Such services involve telephone, circuit switched data transmission services and ISDN. SS7 provides a reliable transport system and therefore it can be used for management and maintenance information transfer in telecommunication networks. SS7 can be used for both circuit related and non-circuit related signalling [33].

2.7 SIGTRAN

Signalling transport (SIGTRAN) provides a transparent transport of message-based signalling protocols over IP networks. SCTP was introduced by SIGTRAN to fulfil the requirements of telephony signalling applications, primarily robustness and timing [34].

SCTP provides several features that are essential for signalling traffic [35]. SCTP is message oriented and it arranges several messages into one or more chunks. In order to reduce the protocol overhead, SCTP bundles several chunks into one SCTP/IP packet. SS7 requires maintaining sequences of related messages in a specific session, i.e. a voice call or an SMS transaction [36]. This is achieved by using multi-streaming which enables transmitting of several independent parallel streams of chunks within one session. A sequence number is assigned to each message in a stream which allows messages to be independently ordered. Another feature is multi-homing that

enables one or both endpoints to have multiple IP addresses and achieves transparent fail-over between redundant network paths.

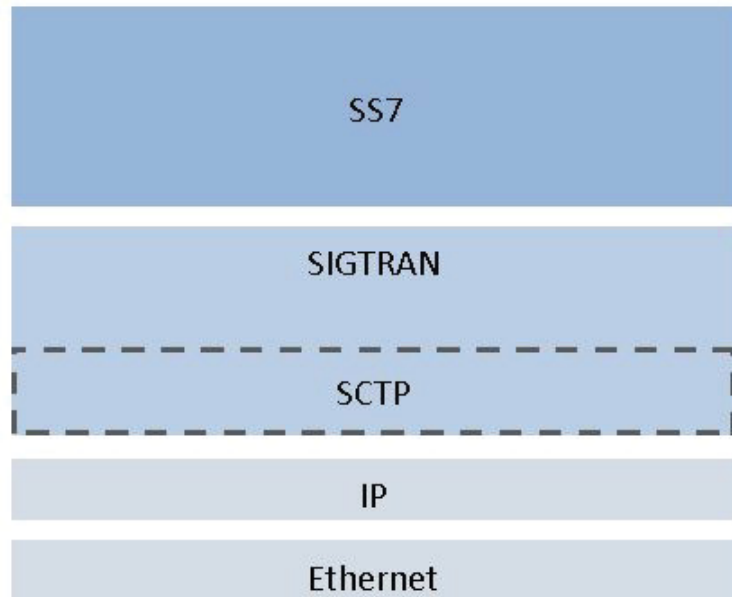


Figure 2.3: SIGTRAN protocol architecture

Chapter 3

Experimental Setup

3.1 Testbed

The testbed used for the research consists of two servers, several simulators and one storage. Both servers are HP DL380 G6 servers. Each server has two Intel Xeon-Core X5550 processors (2.66 GHz, 8 MB L3 Cache, DDR3 1333), 24 GB of RAM (1333 MHz DDR3) and HP Smart Array P410i including four 160 GB hard disks. The storage contains 3 LUNs. Two LUNs are 270 GB in size and each is dedicated to storing data of a specific hypervisor and its related instances. Write/read cache is enabled for both LUNs. The third LUN has 680 GB space and is used for a backup purpose.

The storage is connected to both servers using fiber channel high speed links to prevent any unwanted interruptions or delays. All other components are connected to 1 gigabit backbone via 1 gigabit Ethernet cables.

3.2 Cluster application

The telecommunication services are provided by a cluster application which was supplied by a telecommunication company and installed on the servers. The application works in active and passive mode. All the traffic is handled by the active node of the application and the passive node is used when the active node goes down. A separate interconnect link between the two servers is required for each cluster application in order to check the heartbeat messages of the active node. The nodes of each cluster application use internal communication to maintain awareness of all other nodes in the cluster. This prevents any conflicts between different clusters residing on the same machine in virtual environment. The application reports the self-behaviour including throughput, response time and number of successful and failed requests.

3.3 Simulators

The traffic is generated by simulators which were also provided by a telecommunication company. The traffic is based on Signalling System No. 7 (SS7) protocol [31]. The simulators use SIGTRAN to transmit and convert the SS7 traffic in IP networks. Each simulator can produce a limited traffic load. Therefore several simulators must be utilized to generate a required traffic load, i.e high, moderate and low.

3.4 Measurement tools

Several tools are available for measuring the CPU utilization in KVM, e.g. *top* and *sar* commands. Outputs of the two commands are very different. The output of *sar* is well structured and can be easily imported into MATLAB environment and visualized. Therefore the *sar* command was used to measure CPU utilization in KVM. This command is a part of *sysstat* package and it is capable of monitoring system activities for long periods of time.

CPU measurements in XEN were performed using *xentop* command. *Xentop* is a command that performs measurements of CPU utilization in XEN.

The application provides statistical data about response time and throughput. This approach was suitable thanks to reliability and simplicity of the statistical data sets and it was used to identify the behaviour of telecommunication services.

MATLAB software was used to visualize the raw data and for calculation of mean, variance, standard deviation and margin of error.

3.5 Non-virtualized environment setup

Before assessing the performance in virtual environment, it is necessary to evaluate the performance in non-virtualized scenario. Understanding the behaviour of services in non-virtualized environment helps to identify the impact of virtualization more accurately.

The layout of the testbed is described in section 3.1 and depicted in Figure 3.1. There is a direct link between two servers which is designed for the required communication of the cluster application. The SS7 traffic is separated from the rest by using a different connection. The data is stored in local hard disks on the servers.

3.6 Virtualized environment setup

In this setup, the amount of memory is increased to 48 GB so the memory resources allotted to virtual machines stay the same as in non-virtualized

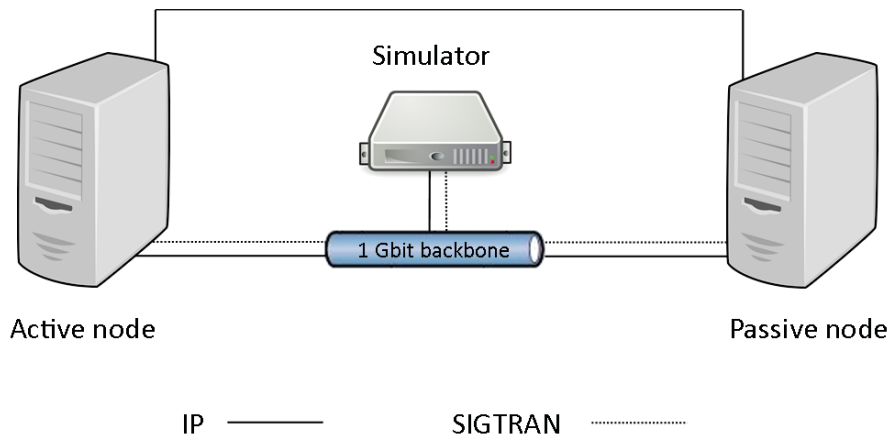


Figure 3.1: Non-virtualized testbed

scenario. Additionally, two virtual machines are created for each server and one direct connection is required for the communication of cluster application on each VM. A separate link is dedicated for migration traffic to avoid exhausting other links. As it has been mentioned before, the storage is divided to separate LUNs corresponding to each hypervisor. Virtual machine monitors, i.e. *virt-manager*, are used for VM administration. The testbed is depicted in the figure 3.2.

3.6.1 KVM

KVM hypervisor is installed during Red Hat Enterprise Server 6.2 (kernel version 2.6.32-220.32.1.el6) installation by selecting the *virtualization package group*. This option contains *KVM hypervisor*, *virt-manager*, *libvirt* and *virt-viewer* (qemu-kvm 2:0.12.1.2-2.209.el6_2.5). Since migration process requires a shared storage, the two servers must be configured as a cluster using GFS2 in order to be able to share the same *EMC* storage LUN. GFS2 is included in *Resilient Storage Add-On* and Red Hat supports the use of GFS2 file systems only as implemented in the *High Availability Add-On*. Bridge networking is used to dedicate a physical network interface to a virtual machine. Para-virtualization is utilized by using *Virtue* drivers for hard disks and network adaptors. Para-virtualized drivers decrease the guest I/O latency and increase throughput to near bare-metal levels. The *virsh* and *virt-manager* programs are the main interfaces for the guest management.

3.6.2 XEN

XEN is installed on top of the SLES 11 SP2 operation system by selecting the XEN kernel (kernel version 3.0.13) and packages during the SLES in-

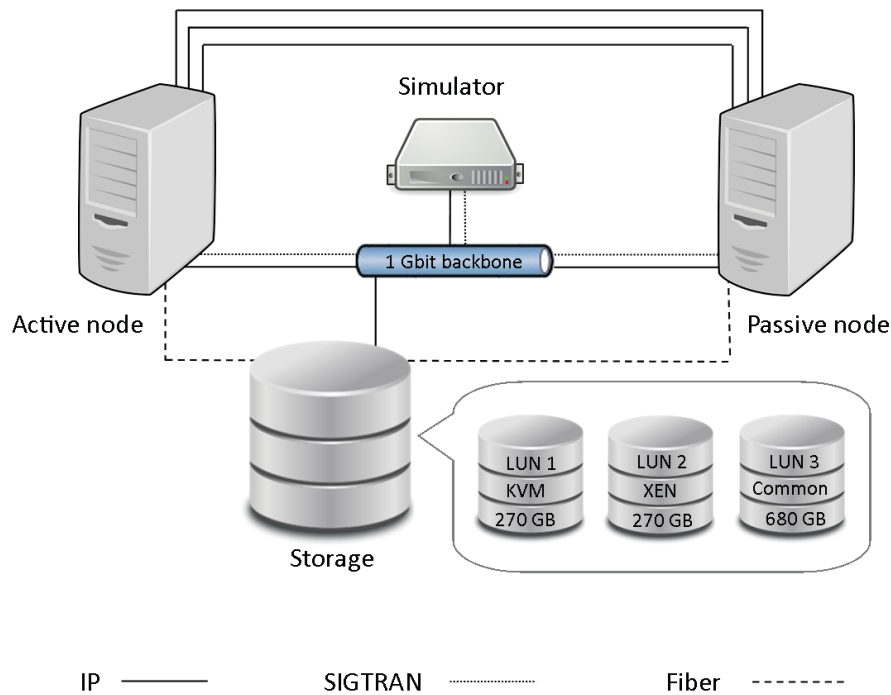


Figure 3.2: Virtualized testbed

stallation. *OCFS2* is the file system which is used to configure the servers as a cluster. It is part of *SUSE Linux Enterprise High Availability Extension*. XEN is using bridging for networking. *PVHVM* drivers (PV-on-HVM drivers) are used as disk and network drivers. The *xm* program is the main interface for managing guest domains and *virt-manager* is the GUI interface.

3.7 Test cases

3.7.1 Measurement procedure

Non-virtualized and virtualized scenario

The maximum traffic load that can be handled by the application had to be determined for the setup in non-virtual environment. Traffic generated by simulator was being gradually increased until failed requests were detected to determine traffic load limits. Various test cases for low, moderate and high traffic loads were defined based on the maximum traffic load. In the virtualized scenario, the application was not able to handle all test cases due to the virtualization overhead. Consequently, the test cases for each setup

in virtual environment were redefined to reflect the traffic limitation. Whole traffic was sent to the active node. In case of the 6 core setup the traffic was split into two equal traffic loads which were sent to the active nodes. Each test case ran for twenty minutes with a constant traffic. The CPU utilization and average response time were measured in ten second intervals. The test cases were repeated two times.

Migration

Four VMs were created, two on each server, forming two clusters with the active nodes on one server and the passive nodes on the other server. 2250 req/s were sent to the active node of one of the clusters in order to load the system and simulate a more realistic scenario. Additionally, the other active node was loaded with 150 req/s traffic and migrated forth and back two times. CPU utilization and maximum response time were measured in ten second intervals during the migration.

Validation

The validity of the results was ensured by performing the following steps:

- All measurements ran for 20 minutes in order to check that the results are stable.
- All measurements were repeated twice in order to check repeatability and stability of the results.
- Statistical analysis of results was carried out in order to validate the results.

3.7.2 Test cases in non-virtualized scenario

The server had dedicated 16 cores of CPU and 24 GB of RAM. The parameters below were measured in ten second intervals during the test.

- CPU utilization (%)
- Average response time (ms)

Table 3.1 demonstrates the test cases that were carried out.

Test case number	Load [req/s]
1.1	750
1.2	2250
1.3	4500
1.4	6450
1.5	7950
1.6	9450

Table 3.1: Test cases in the non-virtualized scenario

3.7.3 Test cases in virtualized scenario

The same test cases were performed for different configurations of CPU cores and memory resources. Lack of CPU resources is the bottleneck in this experiment and thus assessment of the impact of using less CPU cores in VMs was very important.

In the first test case, one VM was created on each server with dedicated 16 cores of CPU and 24 GB of RAM as in the non-virtualized system. Therefore a fair performance comparison of these two scenarios was achieved.

In the previous test case, no CPU core resources were allotted to the hypervisor itself. Hence, in the second test case, 12 CPU cores were dedicated to VMs and 4 cores were retained for the hypervisor's internal use.

The main objective of virtualization is to share resources among several instances to better utilize the resources. In the next test case, 12 CPU cores were equally divided between two VMs on each server (6 cores per each VM) with dedicated 14 GB of RAM. The traffic load was also split between two VMs to achieve an acceptable comparison with the 12 core setup test case. This setup is called the 6 core setup in this report.

All test cases are clearly defined in the sections below.

KVM

Test case number	Load [req/s]	Feasible
2.1.1	750	yes
2.1.2	2250	yes
2.1.3	4500	yes
2.1.4	6450	yes
2.1.5	7950	yes
-----	9450	no

Table 3.2: KVM - Test cases for the 16 core setup

Test case number	Load	Feasible [req/s]
2.2.1	750	yes
2.2.2	2250	yes
2.2.3	4500	yes
2.2.4	6450	yes
2.2.5	7350	yes
	7950	no
	9450	no

Table 3.3: KVM - Test cases for the 12 core setup

Test case number	Load to each VM [req/s]	Total Load [req/s]	Feasible
2.3.1	375	750	yes
2.3.2	1125	2250	yes
2.3.3	2250	4500	yes
2.3.4	3225	6450	yes
2.3.5	3675	7350	yes
2.3.6	4425	8850	yes
	4725	9450	no

Table 3.4: KVM - Test cases for the 6 core setup

XEN

Test case number	Load [req/s]	Feasible
2.4.1	750	yes
2.4.2	2250	yes
2.4.3	4500	yes
2.4.4	6450	yes
	7950	no
	9450	no

Table 3.5: XEN - Test cases for the 16 core setup

Test case number	Load [req/s]	Feasible
2.5.1	750	yes
2.5.2	2250	yes
2.5.3	4500	yes
2.5.4	5700	yes
-----	6450	no
	7950	no
	9450	no

Table 3.6: XEN - Test cases for the 12 core setup

Test case number	Load to each VM [req/s]	Total Load [req/s]	Feasible
2.6.1	375	750	yes
2.6.2	1125	2250	yes
2.6.3	2250	4500	yes
2.6.4	3225	6450	yes
-----	3975	7950	no
	4725	9450	no

Table 3.7: XEN - Test cases for the 6 core setup

3.7.4 Migration test cases

Downtime and total migration time measurements were the main purpose of the migration test cases [37]. Moreover, understanding the system characteristics such as CPU utilization and response time during the migration period is crucial since a high response time is not acceptable in telecommunication applications. Furthermore, adequate CPU resources assure a good QoS whereas their shortage may cause difficulties in handling the services. Four VMs were created, two on each server. They formed two clusters, with one VM on each server. As it is depicted in 3.3, 2250 req/s were sent to one of the clusters and the other one was loaded with 150 req/s. The active node, which was handling 150 req/s, was migrated back and forth.

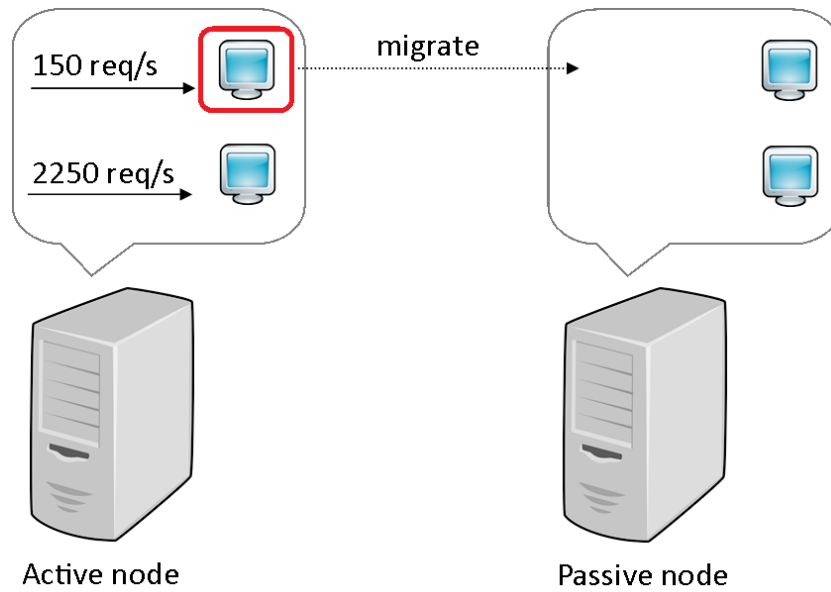


Figure 3.3: Migration test case

The total migration time is dependent on the number of dirty pages [25]. Dirty pages are increasing over the time when VMs are running. In order to have an equivalent situation for all migration test cases, the VMs had been running for fifteen minutes before the migration started. Following parameters were measured during the migration:

- Downtime: Downtime of VMs which can cause a delay or downtime in services, failed requests in this case.
- Total migration time
- CPU utilization
- Maximum response time

Chapter 4

Results

4.1 Virtual vs. non-virtual scenario

A comparison of virtualized and non-virtualized environments is an important part of the research. Since virtualization affects performance of the system, it is vital to investigate the influence of virtualization on behaviour of the services.

4.1.1 KVM

CPU utilization

Figure 4.1 shows CPU utilization for different core setups. As it can be seen, CPU utilizations differ depending on the number of cores.

While increasing the traffic, the CPU utilization is growing almost linearly. It means that the system behaviour is predictable in terms of CPU utilization. On the contrary, the slopes vary for different core setups. System behaviour can be divided into low and high traffic scenarios based on the intersection point. The 16 core setup has poor overall performance. For the low traffic environment, the 12 core setup would be the best replacement for the non-virtualized environment. As regards the high traffic scenario, the 6 core setup has the lowest CPU utilization, even lower than non-virtualized system. Moreover, it is capable of handling more traffic. Therefore it seems better to have several instances with less CPU cores for high traffic loads.

Average response time

Figure 4.2 shows average response times for different core setups.

While increasing the traffic, the average response times are growing exponentially. As it can be observed, the 6 core setup performed the best among

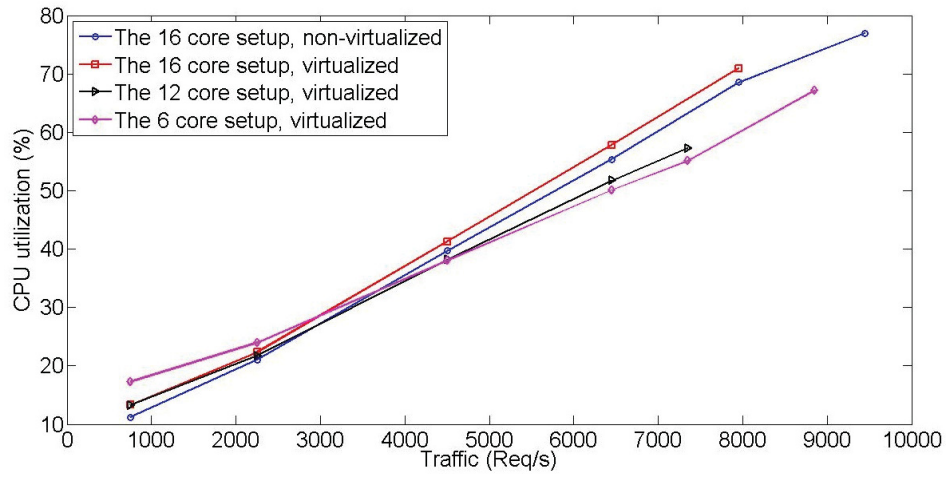


Figure 4.1: KVM - CPU utilization

virtualized scenarios in terms of average response time and handling maximum traffic load.

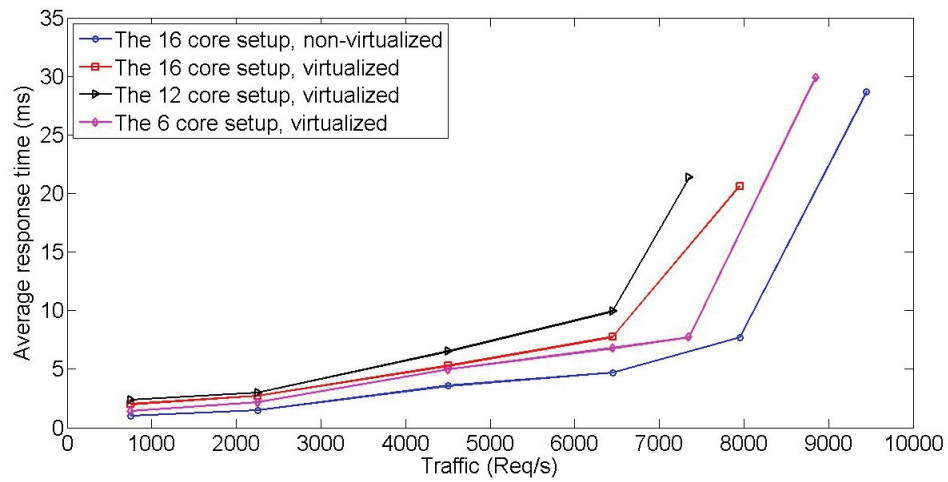


Figure 4.2: KVM - Average response time

4.1.2 XEN

CPU utilization

Figure 4.3 shows CPU utilization for different core setups.

CPU utilization for XEN also shows almost linear pattern. The values are much higher than in the non-virtual scenario. This fact affects the maximum traffic which could be handled by the system. Among the virtual scenarios, there is also an intersection point in CPU utilization but in a higher traffic range than for KVM.

The 12 and the 16 core setups had quite similar behaviour. However, the 16 core setup would be a better replacement for non-virtualized environment since it performed better in terms of maximum traffic and the CPU utilization difference is insignificant.

The 6 core setup has a really high CPU utilization and it would only be a good option for high traffic loads because it shows less CPU utilization in comparison with the 16 core setup.

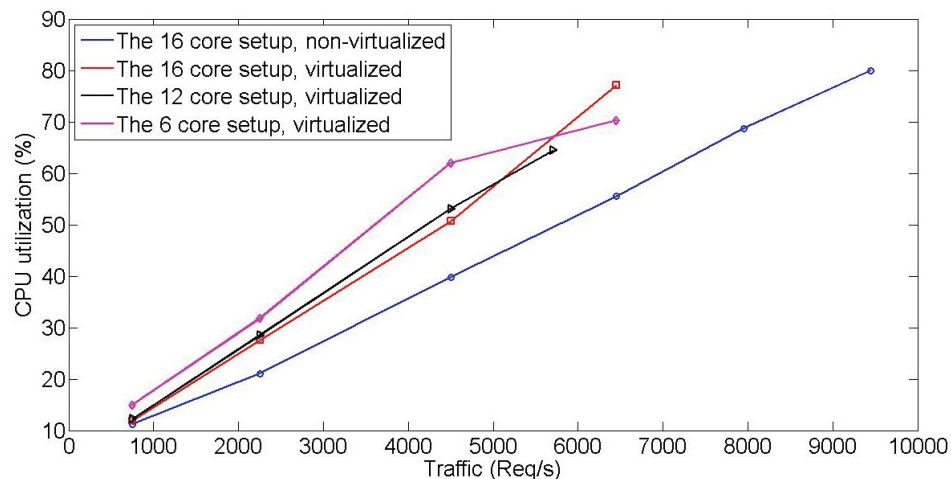


Figure 4.3: XEN - CPU utilization

Average response time

Figure 4.4 shows average response times measured for different core setups. Average response times follow exponential pattern as the traffic increases. The 16 core setup has the best performance among virtualized scenarios in terms of average response time. The 12 core setup has a better response time in low traffic while the 6 core setup performed better in a high traffic. However, non-virtualized system had always lower response time.

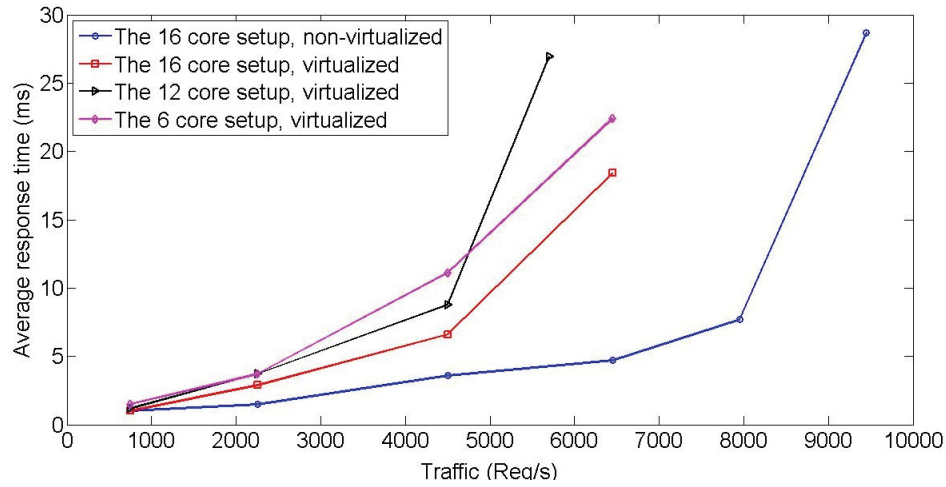


Figure 4.4: XEN - Average response time

4.2 Migration scenario

4.2.1 KVM

CPU utilization

Figure 4.5 shows the CPU utilization during the migration process. As it can be seen, migration adds approximately 8% CPU overhead on the active side and around 3% overhead on the passive side.

Maximum response time

Figure 4.6 depicts the maximum response time during the migration process. It is important to monitor the maximum response time during the migration process since it allows to determine the downtime of the system. The system always responded in less than 20 ms as long as the virtual machine was not moving from one server to the other. Figure 4.6 shows that system downtime is between 650 and 700 ms.

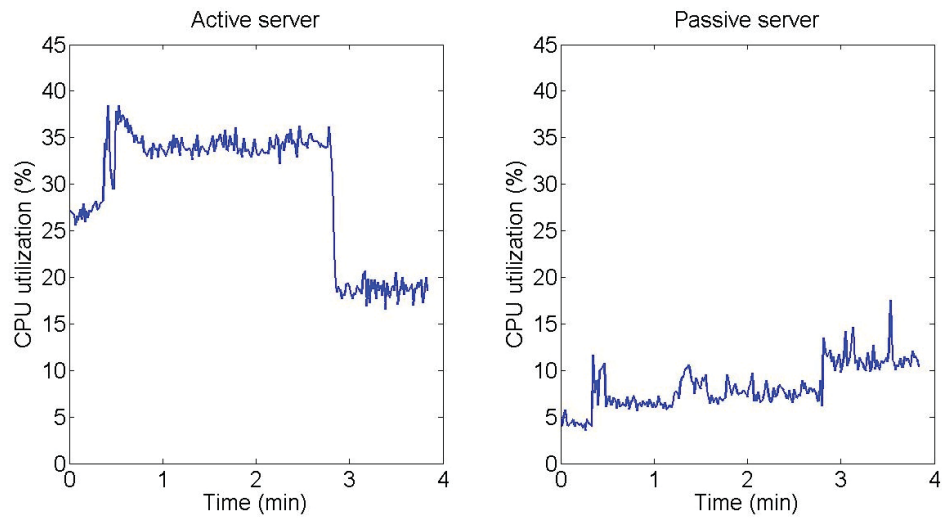


Figure 4.5: KVM - CPU utilization during the migration process

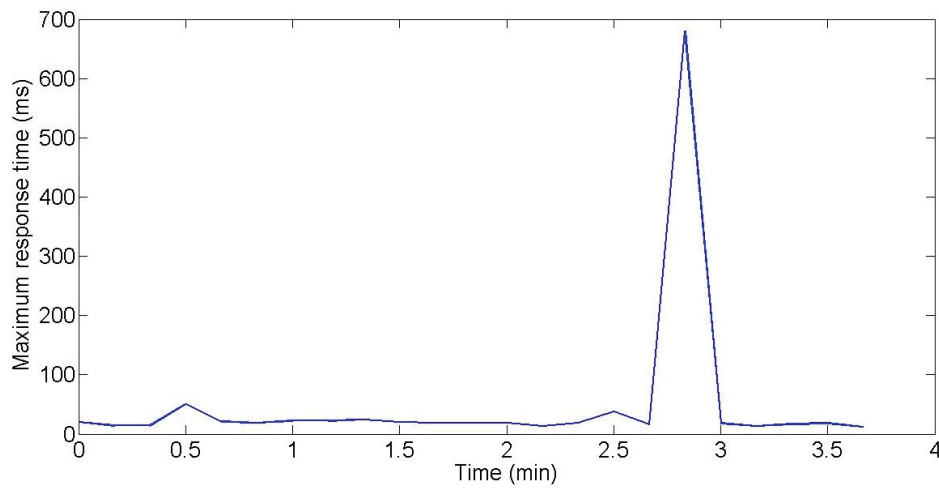


Figure 4.6: KVM - Maximum response time during the migration process

Total migration time and downtime

The migration took about 2 to 3 minutes during this test case. As it is shown in figure 4.6, the value of system downtime is between 650 ms and 700 ms for KVM.

4.2.2 XEN

CPU utilization

CPU utilization during the migration process is depicted in figure 4.7. On average, the CPU overhead of the active side is around 6% while the passive side had 4% overhead.

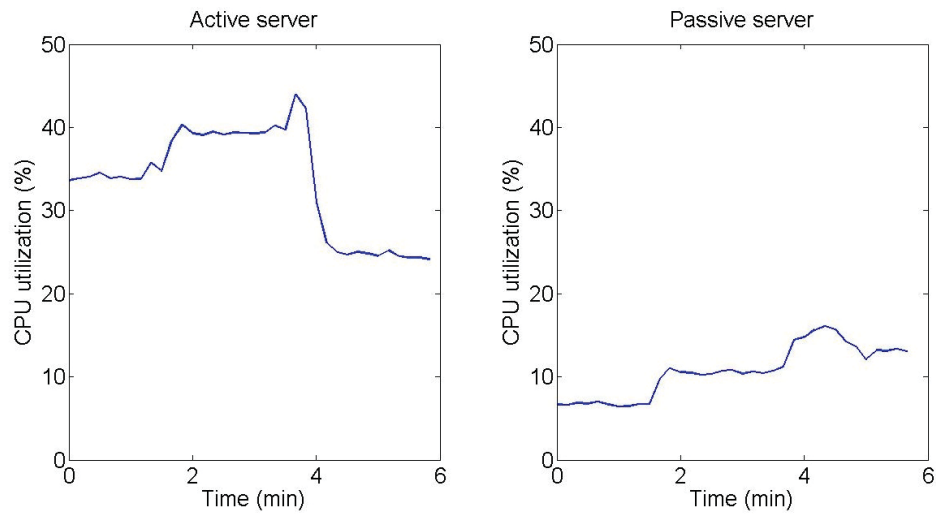


Figure 4.7: XEN - CPU utilization during the migration process

Maximum response time

Figure 4.8 shows the maximum response time during the migration process. Arithmetic average of the maximum response time of the system is less than 20 ms. The value of system downtime is between 250 and 300 ms during the movement of the VM from the active side to the passive side.

Total migration time and downtime

As it is ascertained from the tests, the total migration time in XEN is around two and half minutes and the downtime is between 250 and 300 ms.

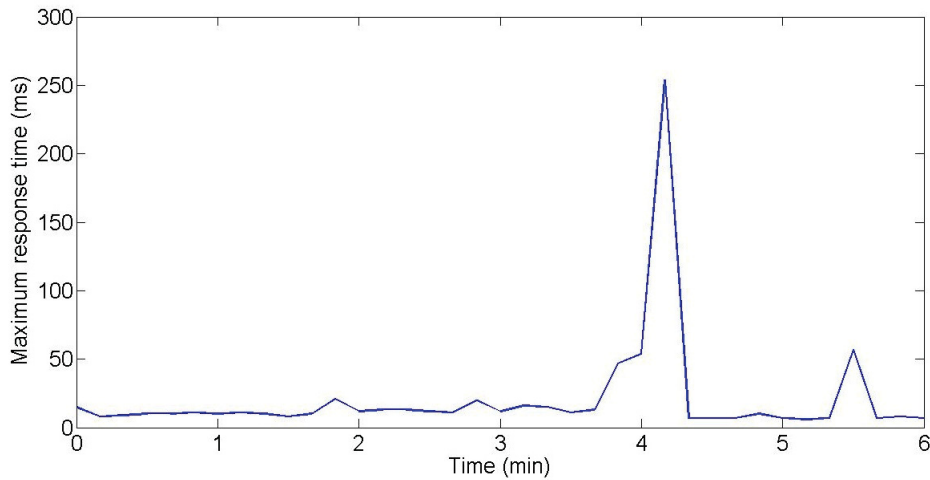


Figure 4.8: XEN - Maximum response time during the migration process

4.3 Result summary

Since the application is CPU-dependent, availability of CPU resources becomes more vital for higher traffic loads. Moreover, the system also must perform handling and scheduling of the CPU resources simultaneously. The experimental measurements have shown that the 6 core setup provides the best performance for high traffic. This is most probably caused by less fight over the CPU resources.

KVM showed a higher CPU overhead during migration and also suffered of a significantly higher downtime in comparison to XEN.

Differences in performance behaviour between the two hypervisors were clearly noticeable. The significance of this finding lies in identifying strengths and weaknesses of each hypervisor. Thus telecommunication companies can use this information to select and employ the most suitable hypervisor based on the requirements.

Results of the experiments are summarized below. Three traffic loads were selected from middle of the traffic spectrum since the system performs better and is more stable in that range.

The following tables provide the comparison of results.

4.3.1 Non-virtualized vs virtualized environments

	2000 [req/s]	3500 [req/s]	5000 [req/s]
the 16 core setup, non-virtualized	19%	32%	42%
the 16 core setup, KVM	22%	35%	48%
the 16 core setup, XEN	25%	41%	58%

Table 4.1: Non-virtualized vs virtualized environment - CPU utilization

	2000 [req/s]	3500 [req/s]	5000 [req/s]
the 16 core setup, non-virtualized	1,5ms	3ms	4,5ms
the 16 core setup, KVM	3ms	4ms	7ms
the 16 core setup, XEN	4ms	9ms	14ms

Table 4.2: Non-virtualized vs virtualized environment - response time

4.3.2 Various CPU core setups

	2000 [req/s]	3500 [req/s]	5000 [req/s]
the 16 core setup	22%	35%	48%
the 12 core setup	21%	31%	43%
the 6 core setup	24%	33%	42%

Table 4.3: KVM - CPU utilization overview

	2000 [req/s]	3500 [req/s]	5000 [req/s]
the 16 core setup	25%	41%	58%
the 12 core setup	25%	42%	58%
the 6 core setup	29%	50%	65%

Table 4.4: XEN - CPU utilization overview

	2000 [req/s]	3500 [req/s]	5000 [req/s]
the 16 core setup	3ms	4ms	7ms
the 12 core setup	3ms	5ms	8ms
the 6 core setup	2ms	4ms	6ms

Table 4.5: KVM - Response time overview

	2000 [req/s]	3500 [req/s]	5000 [req/s]
the 16 core setup	3ms	6ms	10ms
the 12 core setup	4ms	8ms	17ms
the 6 core setup	4ms	9ms	14ms

Table 4.6: XEN - Response time overview

4.3.3 Live migration

	CPU overhead	Downtime	Total migration time
the 6 core setup, KVM	8%	650-700ms	120-180s
the 6 core setup, XEN	6%	250-300ms	120-150s

Table 4.7: Live migration measurements

4.4 Statistical analysis

4.4.1 Formulas

Mean

The formula which was used for calculating mean is as follows

$$\bar{X} = \frac{(\sum X_i)}{N},$$

where

\bar{X} is mean,

$\sum X_i$ is sum of all data values,

N is number of all data values.

Variance

The formula which was used for calculating variance is as follows

$$\sigma^2 = \frac{\Sigma (X_i - \bar{X})^2}{N},$$

where

σ^2 is variance,

\bar{X} is mean,

X_i is each data value,

N is number of all data values.

This formula was used because the data set consists of independent, identically distributed samples.

Standard deviation

The formula which was used for calculating standard deviation is as follows

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\Sigma (X_i - \bar{X})^2}{N}},$$

where

σ is standard deviation,

σ^2 is variance,

\bar{X} is mean,

X_i is each data value,

N is number of all data values.

Margin of error

The formula which was used for calculating margin of error is as follows

$$E = z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{N}},$$

where

E is margin of error,

$z_{\frac{\alpha}{2}}$ is a confidence coefficient,

σ is standard deviation,

N is number of all data values.

Since the calculations were made for a large (more than 30 samples) data set which follows normal distribution and has 95% confidence interval, the confidence coefficient has value 1,96.

4.4.2 Non-virtualized environment

CPU

	Mean	Variance	Standard deviation	Margin of error
750	11.2035	0.0771	0.2776	0.0497
2250	21.0110	0.0814	0.2854	0.0522
4500	39.6910	0.6339	0.7962	0.1455
6450	55.3839	1.0920	1.0450	0.1962
7950	68.6537	0.5117	0.7153	0.1259
9450	79.8549	1.1846	1.0884	0.2133

Table 4.8: CPU data analysis for the non-virtualized environment

Response time

	Mean	Variance	Standard deviation	Margin of error
750	1.0000	0	0	0
2250	1.4809	0.2977	0.5456	0.0934
4500	3.5769	0.4940	0.7029	0.1208
6450	4.7063	0.7691	0.8770	0.1531
7950	7.6918	4.4492	2.1093	0.3422
9450	34.4851	120.5123	10.9778	2.1410

Table 4.9: Response time analysis for the non-virtualized environment

4.4.3 KVM

CPU

Traffic	Mean	Variance	Standard deviation	Margin of error
750	13.2893	0.1470	0.3834	0.0689
2250	22.3508	0.1578	0.3972	0.0726
4500	41.3041	0.6633	0.8144	0.1515
6450	57.7827	3.3610	1.8333	0.3558
7950	70.9295	3.4928	1.8689	0.3358

Table 4.10: KVM - CPU data analysis for the 16 core setup

Traffic	Mean	Variance	Standard deviation	Margin of error
750	13.2488	0.1919	0.4381	0.0784
2250	21.7147	0.2183	0.4672	0.0832
4500	39.2921	0.4243	0.6514	0.1196
6450	51.7258	3.0876	1.7572	0.3284
7350	57.2426	2.2010	1.4836	0.2654

Table 4.11: KVM - CPU data analysis for the 12 core setup

Traffic	Mean	Variance	Standard deviation	Margin of error
750	17.2589	0.2019	0.4493	0.0832
2250	23.9801	0.1759	0.4194	0.0763
4500	38.0321	0.9280	0.9633	0.1800
6450	50.1241	0.0855	0.2924	0.0537
7350	55.1593	0.3001	0.5479	0.0972
8850	67.2001	0.2324	0.4821	0.0881

Table 4.12: KVM - CPU data analysis for the 6 core setup

Response time

Traffic	Mean	Variance	Standard deviation	Margin of error
750	2.0000	0	0	0
2250	2.7328	0.1973	0.4442	0.0761
4500	5.2901	0.8537	0.9239	0.1582
6450	7.7302	6.4386	2.5374	0.4431
7950	23.2750	26.5708	5.1547	0.9223

Table 4.13: KVM - Response time data analysis for the 16 core setup

Traffic	Mean	Variance	Standard deviation	Margin of error
750	2.0000	0	0	0
2250	2.9854	0.0733	0.2708	0.0453
4500	6.5000	1.0426	1.0211	0.1755
6450	9.9440	5.4565	2.3359	0.4095
7350	22.2017	38.8234	6.2308	1.1195

Table 4.14: KVM - Response time data analysis for the 12 core setup

Traffic	Mean	Variance	Standard deviation	Margin of error
750	1.3976	0.2401	0.4900	0.0601
2250	2.1603	0.1504	0.3879	0.0470
4500	4.9880	0.6786	0.8238	0.1021
6450	6.7710	3.9397	1.9849	0.2403
7350	7.7240	3.7905	1.9469	0.2285
8850	33.5574	577.3076	24.0272	3.0720

Table 4.15: KVM - Response time data analysis for the 6 core setup

4.4.4 XEN

CPU

Traffic	Mean	Variance	Standard deviation	Margin of error
750	11.8787	0.1154	0.3397	0.0615
2250	27.5136	0.2013	0.4486	0.0813
4500	50.6293	0.1428	0.3779	0.0703
6450	77.0655	0.6056	0.7782	0.1410

Table 4.16: XEN - CPU data analysis for the 16 core setup

Traffic	Mean	Variance	Standard deviation	Margin of error
750	12.1938	0.0906	0.3010	0.0568
2250	28.4704	0.1258	0.3547	0.0669
4500	53.0746	0.3595	0.5996	0.1126
5700	64.4132	0.3133	0.5597	0.1092

Table 4.17: XEN - CPU data analysis for the 12 core setup

Traffic	Mean	Variance	Standard deviation	Margin of error
750	14.9203	0.1805	0.4248	0.0801
2250	31.7715	1.1594	1.0767	0.1880
4500	61.9979	5.4512	2.3348	0.4344
6450	70.2605	2.1911	1.4802	0.2766

Table 4.18: XEN - CPU data analysis for the 6 core setup

Response time

Traffic	Mean	Variance	Standard deviation	Margin of error
750	1.0000	0	0	0
2250	2.8815	0.1799	0.4241	0.0715
4500	6.6172	1.0413	1.0204	0.1768
6450	18.4148	32.8714	5.7334	0.9672

Table 4.19: XEN - Response time data analysis for the 16 core setup

Traffic	Mean	Variance	Standard deviation	Margin of error
750	1.1520	0.1299	0.3605	0.0632
2250	3.7200	0.4452	0.6672	0.1170
4500	8.7717	3.0824	1.7557	0.3053
5700	31.2642	49.7772	7.0553	1.3431

Table 4.20: XEN - Response time data analysis for the 12 core setup

Traffic	Mean	Variance	Standard deviation	Margin of error
750	1.4512	0.2486	0.49861	0.0623
2250	3.7183	0.4009	0.6332	0.0736
4500	8.4922	7.0666	2.6583	0.3256
6450	22.4040	110.2658	10.5008	1.3017

Table 4.21: XEN - Response time data analysis for the 6 core setup

Chapter 5

Conclusions

This master thesis presents a comparison study of different virtualization technologies and their impacts on the performance of telecommunication services. Performance tests were conducted in both non-virtualized and virtualized environments in order to investigate the effects of virtualization. Moreover, various testbed configurations were used to clearly distinguish which of the hypervisors better complies with the requirements of the telecommunication industry.

Four research questions were formulated:

1. How does virtualization impact the response time and CPU utilization of telecommunication services?

In order to obtain comparable statistical data from both virtualized and non-virtualized systems, the same number of CPU cores and RAM was used during the tests. As it can be seen from tables 4.1 and 4.2, virtualization adds overheads to both CPU utilization and response time.

2. Which hypervisor has better performance in terms of migration, response time and CPU utilization?

KVM demonstrated better performance in terms of CPU utilization and response time. Therefore KVM would be a better choice for telecommunication services sensitive to CPU resources and response time. On the contrary, XEN has shown less downtime in comparison to KVM which makes it more suitable for large environments where maintenance, fault-tolerance and manageability are essential.

3. Are there any significant differences in the performance of hypervisors?

In general, high overheads were measured in XEN for all setups, while overhead values observed in KVM were insignificant. An obvious difference between KVM and XEN behaviour was observed during the 6 core setup tests. Overall, the 6 core setup had the best performance among all setups using KVM, whilst it has the worst performance in XEN.

4. How does performance of the hypervisors differ for various test cases defined in section 3.7 of Chapter 3?

Two scenarios were used for comparison of different test cases. First scenario is comparison of the 16 and 12 core setups which reflects the impact of different number of allocated CPU cores in VMs. Second is the comparison of the 12 and 6 core setups which shows the effect of using multiple VMs with a constant number of allocated CPU resources.

Analysis of the test results have shown that decreasing number of allocated CPU cores causes a higher response time in both hypervisors. Although CPU utilization in XEN was not noticeably affected, considerably less CPU utilization was observed in KVM.

Using multiple VMs in KVM decreased the response time and CPU overhead. However, running the same tests in XEN caused increased CPU utilization. Additionally, response time was shorter for high traffic loads and it was increased for low traffic loads.

5.1 Future work

It would be very compelling to continue this research for other hypervisors such as VMWare and HyperV. The results of the additional tests would show how different CPU resource scheduling and network configurations are implemented in the hypervisors. Particular implementation can have a direct effect on performance of telecommunication services.

Since the performance of the services is considerably improved by using multiple VMs in KVM, it would be very interesting to increase the number of VMs on each server. By repeating the tests again, it would be possible to investigate potential virtualization benefits in terms of CPU utilization. Another interesting experiment could be focused on identification of the maximum number of simultaneously running VMs that would allow keeping the system stable from CPU and network resources point of view.

Appendix A

Commands

A.1 CPU utilization

A.1.1 Non-virtualized environment

```
# sar -u 10 > CPU_750_Non_virt.txt
```

A.1.2 KVM

```
# sar -u 10 > CPU_750_KVM_16_core.txt
```

A.1.3 XEN

```
# xentop -b -d 10 > CPU_750_XEN_16_core.txt
```

A.2 Live migration

A.2.1 KVM

```
# date +%x_%H:%M:%S:%N > duration.txt
# virsh migrate --live --verbose dl380x1561-1
  qemu+ssh://root@<IP_Address>/system --migrateuri
  tcp://<IP_Address>:49155
# date +%x_%H:%M:%S:%N >> duration.txt
```

A.2.2 XEN

```
# date +%x_%H:%M:%S:%N > duration.txt
# xm migrate --live dl380x1561-1 <IP_Address>
# date +%x_%H:%M:%S:%N >> duration.txt
```

Appendix B

MATLAB code samples

B.1 CPU utilization

B.1.1 Non-virtualized environment

```
Linux 2.6.32-220.23.1.el6.x86_64 (dl380x1528) 04/25/2013 _x86_64_ (16 CPU)
03:24:41 PM CPU %user %nice %system %iowait %steal %idle
03:24:51 PM all 4.74 0.00 1.05 4.97 0.00 89.25
03:25:01 PM all 5.46 0.00 1.23 4.53 0.00 88.78
03:25:11 PM all 5.44 0.00 1.27 4.76 0.00 88.52
03:25:21 PM all 5.45 0.00 1.22 4.81 0.00 88.51
03:25:31 PM all 5.45 0.00 1.24 4.69 0.00 88.62
03:25:41 PM all 5.49 0.00 1.32 4.61 0.00 88.58
03:25:51 PM all 5.43 0.00 1.23 4.79 0.00 88.54
```

Figure B.1: A sample from CPU data file for non-virtualized environment

MATLAB code

```
fid = fopen('CPU_750.Non_virt.txt');
CPU_750 = textscan(fid, '%s %s %s %f %f %f %f %f');
fclose(fid);

fid = fopen('CPU_2250.Non_virt.txt');
CPU_2250 = textscan(fid, '%s %s %s %f %f %f %f %f');
fclose(fid);

fid = fopen('CPU_4500.Non_virt.txt');
CPU_4500 = textscan(fid, '%s %s %s %f %f %f %f %f');
fclose(fid);

fid = fopen('CPU_6450.Non_virt.txt');
CPU_6450 = textscan(fid, '%s %s %s %f %f %f %f %f');
fclose(fid);

fid = fopen('CPU_7950.Non_virt.txt');
CPU_7950 = textscan(fid, '%s %s %s %f %f %f %f %f');
fclose(fid);
```

```

fid = fopen('CPU_9450.Non_virt.txt');
CPU_9450 = textscan(fid, '%s %s %s %f %f %f %f %f %f');
fclose(fid);

Total_CPU_750.Non_virt = CPU_750{4} + CPU_750{5} + CPU_750{6} ...
+ CPU_750{7} + CPU_750{8};
Total_CPU_2250.Non_virt = CPU_2250{4} + CPU_2250{5} + CPU_2250{6} ...
+ CPU_2250{7} + CPU_2250{8};
Total_CPU_4500.Non_virt = CPU_4500{4} + CPU_4500{5} + CPU_4500{6} ...
+ CPU_4500{7} + CPU_4500{8};
Total_CPU_6450.Non_virt = CPU_6450{4} + CPU_6450{5} + CPU_6450{6} ...
+ CPU_6450{7} + CPU_6450{8};
Total_CPU_7950.Non_virt = CPU_7950{4} + CPU_7950{5} + CPU_7950{6} ...
+ CPU_7950{7} + CPU_7950{8};
Total_CPU_9450.Non_virt = CPU_9450{4} + CPU_9450{5} + CPU_9450{6} ...
+ CPU_9450{7} + CPU_9450{8};

CPU.Non_virt(1) = mean(Total_CPU_750.Non_virt);
CPU.Non_virt(2) = mean(Total_CPU_2250.Non_virt);
CPU.Non_virt(3) = mean(Total_CPU_4500.Non_virt);
CPU.Non_virt(4) = mean(Total_CPU_6450.Non_virt);
CPU.Non_virt(5) = mean(Total_CPU_7950.Non_virt);
CPU.Non_virt(6) = mean(Total_CPU_9450.Non_virt);

CPU_traffic_non_virt(1) = 750;
CPU_traffic_non_virt(2) = 2250;
CPU_traffic_non_virt(3) = 4500;
CPU_traffic_non_virt(4) = 6450;
CPU_traffic_non_virt(5) = 7950;
CPU_traffic_non_virt(6) = 9450;

plot(CPU_traffic_non_virt, CPU.Non_virt, 'b', 'LineWidth', 2);
hold on
xlabel('Traffic loads (Req/s)', 'FontSize', 10)
ylabel('Total CPU utilization (%)', 'FontSize', 10)
title('Total CPU utilization (%)');

```

B.1.2 KVM, the 16 core setup

```

Linux 2.6.32-220.23.1.el6.x86_64 (dl380x1528) 05/25/2013 _x86_64_ (16 CPU)
01:11:02 PM CPU %user %nice %system %iowait %steal %idle
01:11:12 PM all 0.74 0.00 0.40 0.07 0.00 98.79
01:11:22 PM all 3.07 0.00 1.23 0.62 0.00 95.09
01:11:32 PM all 3.05 0.00 1.14 0.59 0.00 95.22
01:11:42 PM all 4.63 0.00 1.51 0.81 0.00 93.05
01:11:52 PM all 5.29 0.00 1.87 1.08 0.00 91.76
01:12:02 PM all 5.05 0.00 1.71 0.97 0.00 92.28
01:12:12 PM all 4.96 0.00 1.64 0.72 0.00 92.68

```

Figure B.2: A sample from CPU data file for KVM

MATLAB code

```

fid = fopen('CPU_750_KVM_16_core.txt');
CPU_750 = textscan(fid, '%s %s %s %f %f %f %f %f %f');
fclose(fid);

fid = fopen('CPU_2250_KVM_16_core.txt');
CPU_2250 = textscan(fid, '%s %s %s %f %f %f %f %f %f');
fclose(fid);

fid = fopen('CPU_4500_KVM_16_core.txt');
CPU_4500 = textscan(fid, '%s %s %s %f %f %f %f %f %f');
fclose(fid);

fid = fopen('CPU_6450_KVM_16_core.txt');
CPU_6450 = textscan(fid, '%s %s %s %f %f %f %f %f %f');
fclose(fid);

fid = fopen('CPU_7950_KVM_16_core.txt');
CPU_7950 = textscan(fid, '%s %s %s %f %f %f %f %f %f');
fclose(fid);

Total_CPU_750_KVM_16_core = CPU_750{4} + CPU_750{5} + CPU_750{6} ...
+ CPU_750{7} + CPU_750{8};
Total_CPU_2250_KVM_16_core = CPU_2250{4} + CPU_2250{5} + CPU_2250{6} ...
+ CPU_2250{7} + CPU_2250{8};
Total_CPU_4500_KVM_16_core = CPU_4500{4} + CPU_4500{5} + CPU_4500{6} ...
+ CPU_4500{7} + CPU_4500{8};
Total_CPU_6450_KVM_16_core = CPU_6450{4} + CPU_6450{5} + CPU_6450{6} ...
+ CPU_6450{7} + CPU_6450{8};
Total_CPU_7950_KVM_16_core = CPU_7950{4} + CPU_7950{5} + CPU_7950{6} ...
+ CPU_7950{7} + CPU_7950{8};

CPU_KVM_16_core(1) = mean(Total_CPU_750_KVM_16_core);
CPU_KVM_16_core(2) = mean(Total_CPU_2250_KVM_16_core);
CPU_KVM_16_core(3) = mean(Total_CPU_4500_KVM_16_core);
CPU_KVM_16_core(4) = mean(Total_CPU_6450_KVM_16_core);
CPU_KVM_16_core(5) = mean(Total_CPU_7950_KVM_16_core);

CPU_traffic_KVM_16_core(1) = 750;
CPU_traffic_KVM_16_core(2) = 2250;
CPU_traffic_KVM_16_core(3) = 4500;
CPU_traffic_KVM_16_core(4) = 6450;
CPU_traffic_KVM_16_core(5) = 7950;

plot(CPU_traffic_KVM_16_core, CPU_KVM_16_core, 'r', 'LineWidth', 2);
hold on
xlabel('Traffic loads (Req/s)', 'FontSize', 10)
ylabel('Total CPU utilization (%)', 'FontSize', 10)
title('Total CPU utilization (%)');

```

B.1.3 XEN, the 16 core setup

NAME	STATE	CPU(sec)	CPU(%)	MEM(k)	MEM(%)	MAXMEM(k)	MAXMEM(%)	VCPUS	NETS	NETTX(k)	NETRX(k)	VBDS	VBD_00
VBD_RD	VBD_WR	VBD_RSECT	VBD_WSECT	SSID									
d1380x1561	--b---	149273	34.8	24580068	48.8	24580096	48.8	16	4	62167321	199523952	1	
0	135642	79120220	10278820	788315642	0	4.1	no limit	n/a	16	0	0	0	0
Domain-0	-----r	26653	13.1	2058752	4.1	no limit	n/a	16	0	0	0	0	0
0	0	0	0	0	0								
NAME	STATE	CPU(sec)	CPU(%)	MEM(k)	MEM(%)	MAXMEM(k)	MAXMEM(%)	VCPUS	NETS	NETTX(k)	NETRX(k)	VBDS	VBD_00
VBD_RD	VBD_WR	VBD_RSECT	VBD_WSECT	SSID									
d1380x1561	--b---	149280	67.0	24580068	48.8	24580096	48.8	16	4	62169243	199529439	1	
0	135642	79127805	10278820	788376322	0	4.1	no limit	n/a	16	0	0	0	0
Domain-0	-----r	26655	19.1	2058752	4.1	no limit	n/a	16	0	0	0	0	0
0	0	0	0	0	0								
NAME	STATE	CPU(sec)	CPU(%)	MEM(k)	MEM(%)	MAXMEM(k)	MAXMEM(%)	VCPUS	NETS	NETTX(k)	NETRX(k)	VBDS	VBD_00
VBD_RD	VBD_WR	VBD_RSECT	VBD_WSECT	SSID									
d1380x1561	-----r	149288	81.2	24580068	48.8	24580096	48.8	16	4	62171540	199536349	1	
0	135620	79136392	10280244	788445122	0	4.1	no limit	n/a	16	0	0	0	0
Domain-0	-----r	26657	21.2	2058752	4.1	no limit	n/a	16	0	0	0	0	0
0	0	0	0	0	0								

Figure B.3: A sample from CPU data file for XEN

MATLAB code

```

fid = fopen('CPU_750_XEN_16_core.txt');
CPU_750 = textscan(fid, '%s %s %d %f %d %f %d %f %d %d %d %d ...
%d %d %d %d %d %d %d');
fclose(fid);

fid = fopen('CPU_2250_XEN_16_core.txt');
CPU_2250 = textscan(fid, '%s %s %d %f %d %f %d %f %d %d %d %d ...
%d %d %d %d %d %d %d');
fclose(fid);

fid = fopen('CPU_4500_XEN_16_core.txt');
CPU_4500 = textscan(fid, '%s %s %d %f %d %f %d %f %d %d %d %d ...
%d %d %d %d %d %d %d');
fclose(fid);

fid = fopen('CPU_6450_XEN_16_core.txt');
CPU_6450 = textscan(fid, '');
fclose(fid);

j=1;
for i=1:2:length(CPU_750{4})-1
CPU_750_All(j) = CPU_750{4}(i) + CPU_750{4}(i+1);
j = j + 1;
end

j=1;
for i=1:2:length(CPU_2250{4})-1
CPU_2250_All(j) = CPU_2250{4}(i) + CPU_2250{4}(i+1);
j = j + 1;
end

j=1;
for i=1:2:length(CPU_4500{4})-1
CPU_4500_All(j) = CPU_4500{4}(i) + CPU_4500{4}(i+1);
j = j + 1;
end

```

```
j=1;
for i=1:2:length(CPU_6450{4})-1
CPU_6450_All(j) = CPU_6450{4}(i) + CPU_6450{4}(i+1);
j = j +1;
end

Total_CPU_750_XEN_16_core = (CPU_750_All*100)/1600;
Total_CPU_2250_XEN_16_core = (CPU_2250_All*100)/1600;
Total_CPU_4500_XEN_16_core = (CPU_4500_All*100)/1600;
Total_CPU_6450_XEN_16_core = (CPU_6450_All*100)/1600;

CPU_XEN_16_core(1) = mean(Total_CPU_750_XEN_16_core);
CPU_XEN_16_core(2) = mean(Total_CPU_2250_XEN_16_core);
CPU_XEN_16_core(3) = mean(Total_CPU_4500_XEN_16_core);
CPU_XEN_16_core(4) = mean(Total_CPU_6450_XEN_16_core);

CPU_traffic_XEN_16_core(1) = 750;
CPU_traffic_XEN_16_core(2)= 2250;
CPU_traffic_XEN_16_core(3)= 4500;
CPU_traffic_XEN_16_core(4)= 6450;

plot(CPU_traffic_XEN_16_core,CPU_XEN_16_core,'r','LineWidth',2);
hold on
xlabel('Traffic loads (Req/s)','FontSize',10)
ylabel('Total CPU utilization (%)','FontSize',10)
title('Total CPU utilization (%)');
```



```

fclose(fid);

fid = fopen('RT_9450_Non_virt.txt');
RT_9450 = textscan(fid, '%s %s %f %f %f %f %f %f %f %f');
fclose(fid);

RT_Non_virt(1) = mean(RT_750{8});
RT_Non_virt(2) = mean(RT_2250{8});
RT_Non_virt(3) = mean(RT_4500{8});
RT_Non_virt(4) = mean(RT_6450{8});
RT_Non_virt(5) = mean(RT_7950{8});
RT_Non_virt(6) = mean(RT_9450{8});

RT_traffic_non_virt(1) = 750;
RT_traffic_non_virt(2) = 2250;
RT_traffic_non_virt(3) = 4500;
RT_traffic_non_virt(4) = 6450;
RT_traffic_non_virt(5) = 7950;
RT_traffic_non_virt(6) = 9450;

plot(RT_traffic_non_virt, RT_Non_virt, 'b', 'LineWidth', 2);
hold on
xlabel('Traffic loads (Req/s)', 'FontSize', 10)
ylabel('Average Response time (ms)', 'FontSize', 10)
title('Average Response time (ms)');

```

B.2.2 KVM, the 16 core setup

MATLAB code

```

fid = fopen('RT_750_KVM_16_core.txt');
RT_750 = textscan(fid, '%s %s %f %f %f %f %f %f %f %f');
fclose(fid);

fid = fopen('RT_2250_KVM_16_core.txt');
RT_2250 = textscan(fid, '%s %s %f %f %f %f %f %f %f %f');
fclose(fid);

fid = fopen('RT_4500_KVM_16_core.txt');
RT_4500 = textscan(fid, '%s %s %f %f %f %f %f %f %f %f');
fclose(fid);

fid = fopen('RT_6450_KVM_16_core.txt');
RT_6450 = textscan(fid, '%s %s %f %f %f %f %f %f %f %f');
fclose(fid);

fid = fopen('RT_7950_KVM_16_core.txt');
RT_7950 = textscan(fid, '%s %s %f %f %f %f %f %f %f %f');
fclose(fid);

```

Succ	Fail	Retransmission Detected	Duplicate Answered	Thrput (/s)	Response time (ms)		
					Avg	Min	Max
240	0	0	0	23	2	1	25
0	0	0	0	0	-	-	-
240	0	0	0	23	2	1	25
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
240	0	0	0	23	2	1	25
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
1334	0	0	0	133	2	1	23
0	0	0	0	0	-	-	-
1334	0	0	0	133	2	1	23
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
1334	0	0	0	133	2	1	23
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-

Figure B.5: A sample from response time data file for KVM

```

RT_KVM_16_core(1) = mean(RT_750{8});
RT_KVM_16_core(2) = mean(RT_2250{8});
RT_KVM_16_core(3) = mean(RT_4500{8});
RT_KVM_16_core(4) = mean(RT_6450{8});
RT_KVM_16_core(5) = mean(RT_7950{8});

RT_traffic_KVM_16_core(1) = 750;
RT_traffic_KVM_16_core(2) = 2250;
RT_traffic_KVM_16_core(3) = 4500;
RT_traffic_KVM_16_core(4) = 6450;
RT_traffic_KVM_16_core(5) = 7950;

plot(RT_traffic_KVM_16_core, RT_KVM_16_core, 'r', 'LineWidth', 2);
hold on
xlabel('Traffic loads (Req/s)', 'FontSize', 10)
ylabel('Average Response time (ms)', 'FontSize', 10)
title('Average Response time (ms)');

```

B.2.3 XEN, the 16 core setup

Succ	Fail	Retransmission Detected	Duplicate Answered	Thruput (/s)	Response time (ms)		
					Avg	Min	Max
1924	0	0	0	192	1	1	7
0	0	0	0	0	-	-	-
1924	0	0	0	192	1	1	7
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
1924	0	0	0	192	1	1	7
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
2664	0	0	0	266	1	1	8
0	0	0	0	0	-	-	-
2664	0	0	0	266	1	1	8
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
2664	0	0	0	266	1	1	8
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-
0	0	0	0	0	-	-	-

Figure B.6: A sample from response time data file for XEN

MATLAB code

```

fid = fopen('RT_750_XEN_16_core.txt');
RT_750 = textscan(fid, '%s %s %f %f %f %f %f %f %f');
fclose(fid);

fid = fopen('RT_2250_XEN_16_core.txt');
RT_2250 = textscan(fid, '%s %s %f %f %f %f %f %f %f');
fclose(fid);

fid = fopen('RT_4500_XEN_16_core.txt');
RT_4500 = textscan(fid, '%s %s %f %f %f %f %f %f %f');
fclose(fid);

fid = fopen('RT_6450_XEN_16_core.txt');
RT_6450 = textscan(fid, '%s %s %f %f %f %f %f %f %f');
fclose(fid);

RT_XEN_16_core(1) = mean(RT_750{8});
RT_XEN_16_core(2) = mean(RT_2250{8});
RT_XEN_16_core(3) = mean(RT_4500{8});

```

```
RT_XEN_16_core(4) = mean(RT_6450{8});

RT_traffic_XEN_16_core(1) = 750;
RT_traffic_XEN_16_core(2) = 2250;
RT_traffic_XEN_16_core(3) = 4500;
RT_traffic_XEN_16_core(4) = 6450;

plot(RT_traffic_XEN_16_core, RT_XEN_16_core, 'r', 'LineWidth', 2);
hold on
xlabel('Traffic loads (Req/s)', 'FontSize', 10)
ylabel('Average Response time (ms)', 'FontSize', 10)
title('Average Response time (ms)');
```

Bibliography

- [1] Liu, H., Xu, Ch., Jin, H., Gong, J., Liao, X. “Performance and Energy Modeling for Live Migration of Virtual Machines”. *Proceedings of the IEEE International Symposium on High Performance Distributed Computing*, (2011), 171–181.
- [2] Rimal, B.P., Choi, E., Lumb, I. “A taxonomy and survey of cloud computing systems”. *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, (2009), 44–51.
- [3] Younge, A.J., Henschel, R., Brown, J.T., von Laszewski, G., Qiu, J., Fox, G.C. “Analysis of virtualization technologies for high performance computing environments”. *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, (2011), 9–16.
- [4] Jiang, B., Ravindran, B., Kim, Ch. “Lightweight Live Migration for High Availability Cluster Service”. *Proceedings of Stabilization, Safety, and Security of Distributed Systems - 12th International Symposium*, (2010), 420–434.
- [5] Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A. “Live migration of virtual machines”. *Proceedings of the 2nd Symposium on Networked Systems Design & Implementation (NSDI’05)*, (2005), 273–86.
- [6] Svard, P., Hudzia, B., Tordsson, J., Elmroth, E. “Evaluation of Delta Compression Techniques for Efficient Live Migration of Large Virtual Machines”. *SIGPLAN Notices*, (2011), 111–20.
- [7] Gabrielsson, J., Hubertsson, O., Mas, I., Skog, R. “Cloud computing in telecommunications”. *Ericsson Review*, (2010), 7–11.
- [8] Martucci, L.A., Zuccato, A., Smeets, B., Habib, S.M., Johansson, T., Shahmehri, N. “Privacy, Security and Trust in Cloud Computing: The Perspective of the Telecommunication Industry”. *2012 IEEE 9th Int’l Conference on Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC)*, (2012), 627–32.

- [9] Akoush, S., Sohan, R., Rice, A., Moore, A.W., Hopper, A. "Predicting the performance of virtual machine migration". *Proceedings 18th IEEE/ACM International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS 2010)*, (2010), 37–46.
- [10] Che J., Yu Y., Shi C., Lin W. "A Synthetical Performance Evaluation of OpenVZ, Xen and KVM". *2010 IEEE Asia-Pacific Services Computing Conference, APSCC 2010*, (2010), 587–594.
- [11] Kikuchi, S., Matsumoto, Y. "Impact of Live Migration on Multi-tier Application Performance in Clouds". *2012 IEEE 5th International Conference on Cloud Computing, CLOUD*, (2012), 261–268.
- [12] Liu, H., Jin, H., Xu, Ch., Liao, X. "Performance and Energy Modeling for Live Migration of Virtual Machines". *Cluster Computing*, (2013), 249–264.
- [13] Kikuchi, S., Matsumoto, Y. "Performance Modeling of Concurrent Live Migration Operations in Cloud Computing Systems using PRISM Probabilistic Model Checker". *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing (CLOUD 2011)*, (2011), 49-5-6.
- [14] Mahjoub, M., Mdhaffar, A., Halima, R.B., Jmaiel, M. "A comparative study of the current cloud computing technologies and offers". *Proceedings of the 2011 IEEE International Conference on Network/Cloud Computing and Applications (NCCA 2011)*, (2011), 131–4.
- [15] Wang, L., von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., Fu, Ch. "Cloud Computing: a Perspective Study". *New Generation Computing*, (2010), 137–46.
- [16] Gong, Ch., Liu, J., Zhang, Q., Chen, H., Gong, Z. "The Characteristics of Cloud Computing". *2010 39th International Conference on Parallel Processing Workshops (ICPPW)*, (2010), 275–9.
- [17] Kulkarni, G., Gambhir, J., Patil, T., Dongare, A. "A security aspects in cloud computing". *Proceedings of the 2012 IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS)*, (2012), 547–50.
- [18] Zhang, Q., Cheng, L., Boutaba, R. "Cloud computing: state-of-the-art and research challenges". *Journal of Internet Services and Applications*, (2010), 7–18.
- [19] Mell, P., Grance, T. "The NIST Definition of Cloud Computing". *National Institute of Standards and Technology, Gaithersburg, MD 20899-8930*, (2011).

- [20] Che J., He Q., Gao Q., Huang D. "Performance measuring and comparing of virtual machine monitors". *2008 IEEE/IFIP 5th International Conference on Embedded and Ubiquitous Computing, EUC 2008*, (2008), 381–6.
- [21] Lombardi, F., Di Pietro, R. "Secure virtualization for cloud computing". *Journal of Network and Computer Applications*, (2011), 1113–22.
- [22] Li, Y., Li, W., Jiang, C. "A Survey of Virtual Machine System: Current Technology and Future Trends". *2010 Third International Symposium on Electronic Commerce and Security (ISECS 2010)*, (2010), 332–6.
- [23] Matthews, J. N., Hu, W., Hapuarachchi, M., Deshane, T., Dimatos, D., Hamilton, G., McCabe, M., Owens, J. "Quantifying the performance isolation properties of virtualization systems". *Proceedings of the 2007 Workshop on Experimental Computer Science*, (2007).
- [24] Celesti, A., Tusa, F., Villari, M., Puliafito, A. "Improving Virtual Machine Migration in Federated Cloud Environments". *First International Conferences on Access Networks, Services and Technologies (ACCESS 2010)*, (2010), 61–7.
- [25] Strunk, A. "Costs of Virtual Machine Live Migration: A Survey". *2012 IEEE Eighth World Congress on Services*, (2012), 323–9.
- [26] Voorsluys, W., Broberg, J., Venugopal, S., Buyya, R. "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation". *Cloud Computing. First International Conference, CloudCom 2009*, (2009), 254–65.
- [27] McCoyd, M., Krug, R.B., Goel, D., Dahlin, M., Young, W. "Building a Hypervisor on a Formally Verifiable Protection Layer". *Proceedings of the 2013 46th Hawaii International Conference on System Sciences*, (2013), 5069–78.
- [28] Yokoyama, D., Dias, V., Kloh, H., Bandini, M., Porto, F., Schulze, B., Mury, A. "The impact of Hypervisor layer on database applications". *2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing (UCC 2012)*, (2012), 247–54.
- [29] YamunaDevi, L., Aruna, P., Sudha, D.D., Priya, N. "Security in Virtual Machine Live Migration for KVM". *2011 International Conference on Process Automation, Control and Computing (PACC 2011)*, (2011).
- [30] Petrovic, D, Schiper, A. "Implementing virtual machine replication: A case study using Xen and KVM". *International Conference on Advanced Information Networking and Applications, AINA*, (2012), 73–80.

- [31] Ahamed, S.S.R. “Performance analysis of signaling system 7(SS7) network and protocols to provide call control, remote network management and maintenance capabilities”. *Journal of Applied Information Technology*, (2009), 136–42.
- [32] Moore, T., Kosloff, T., Keller, J., Manes, G., Shenoi, S. “Signaling system 7 (SS7) network security”. *Midwest Symposium on Circuits and Systems*, (2002), 496–9.
- [33] ITU-T Q.700, (1993, March). In *International Telecommunication Union*. Retrieved October 15, 2013, from <http://www.itu.int/rec/T-REC-Q.700-199303-I/e>.
- [34] Chukarin, A., Pershakov, N., Samouylov, K. “Performance of sigtran-based signaling links deployed in mobile networks”. *Proceedings of the 9th International Conference on Telecommunications*, (2007), 163–166.
- [35] Tan, W.-Y., Xu D.-W., Chen W. “Research on the transport performance of SCTP”. *Proceedings of the 2009 International Symposium on Computer Network and Multimedia Technology*, (2009), 3–6.
- [36] Eklund, J., Grinnemo, K.-J., Baucke, S., Brunstrom, A. “Tuning SCTP failover for carrier grade telephony signaling”. *Computer Networks*, (2010), 133–49.
- [37] Weining, L., Tao, F. “Live migration of virtual machine based on recovering system and CPU scheduling”. *Proceedings - 2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference, ITAIC 2011* (2011), 303–307.