A REVIEW ON MONITORING CLOUD PERFORMANCE USING LINPACK BENCHMARK ON KVM IN CLOUDSTACK PLATFORM

[1]Navdeep Kaur , [2]Mr.Harinderpal Singh

[1]M.Tech Student , IT , Adesh Institute Of Engg. &Technology , Faridkot , PTU , (India)
[2]Assistant Professor , CSE , Adesh Institute Of Engg. &Technology ,Faridkot , PTU , (India)

ABSTRACT

Cloud computing is a general term for anything that can be accessed as the service over the internet. From the technical point of view, two pillars of cloud computing are service-oriented architecture and virtualization. The important progress seen lately in virtualization is due to the development of several virtual machine hypervisors. Virtual Machine technology enables multiple OS environments to coexist on the same physical computer in strong isolation from each other. To evaluate the performance of the Cloud infrastructure, I have selected a scientific computing application and well known synthetic benchmark, the Linpack numerical library. Using this application I will able to evaluate the impact in performance of several key factors in cloud computing. This review paper disuses and explores all the necessary concepts and parameters associated with performance evaluation of cloud infrastructure such as the CloudStack Platform, Hypervisor, HyperThreading, Virtual machine, KVM and the LinPack parallel scientific benchmark. The virtual machines have been managed through the KVM hypervisor included in the CloudStack platform.

Keywords: Cloud Computing, CloudStack, Hypervisor, HyperThreading, KVM, LinPack

I INTRODUCTION

Cloud computing is a general term for anything that can be accessed as the service over the internet. The services can be Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Storage as a service [1]. The key properties of the cloud computing is user centric, task centric, powerful, accessible, intelligent and programmable. However, at the same time there are some major obstacles that need to be tackled, such as data transfer bottlenecks due to the more and more data-intensive applications or performance unpredictability issues. From the technical point of view, two pillars of cloud computing are service-oriented architecture (SOA) and virtualization of hardware and software [2, 3]. The important progress seen lately in virtualization is due to the development of several virtual machine hypervisors.

IJARSE

Because of such interest there has been substantial development of open source Cloud management platforms such as CloudStack [1], Open Nebula [2] or Eucalyptus [3]. These platforms provide support for several hypervisors like KVM [4] or Xen [5]. In order to test a platform based on CloudStack, I have selected a scientific computing application and well known synthetic benchmark, the LinPack numerical library. Using this application I will evaluate the impact in performance of several key factors in cloud computing: the number of virtual machines per host, the impudence of the hyper threading and the hard disk I/O. To manage the virtualized services I have utilized the CloudStack platform. This paper is organized as follows. Section II describes the main features of the CloudStack platform. Section III discuses the concept of hypervisor and HyperThreading in Cloud environment. Section IV discuses Virtual machine and explores KVM. The parallel scientific application used as benchmark tests are introduced in Section V.

II CLOUDSTACK PLATFORM

CloudStack is an open source and multi-tenanted cloud orchestration platform for delivering Infrastructure-as-a-Service (IaaS) in cloud computing environments. CloudStack was initially developed by Cloud.com and turned over to the Apache Software Foundation in 2012 with the code available under the Apache 2.0 license [7]. CloudStack is used:

1. To help managed service providers.
2. To create and operate public cloud, private cloud and hybrid IaaS clouds by pooling computing resources with capabilities equivalent to Amazon's Elastic Compute Cloud (Amazon EC2).
3. To manage computing, networking and storage resources.

CloudStack competes with OpenStack and Eucalyptus in the cloud computing management platform market. CloudStack supports multiple virtualization platforms such as Citrix XenServer, VMware vSphere and KVM on Ubuntu or CentOS [7, 8].

2.1 CloudStack Features

1. CloudStack manage multiple geographically distributed data centers.
2. CloudStack API gives full access to all managed resources which makes command line tools easier.
3. CloudStack has Multi-node installation support and load balancing.
4. CloudStack has capability of MySQL replication which is useful for maintaining high availability.
5. Many Service providers and organizations use CloudStack to set up an elastic and on-demand IaaS.
6. CloudStack can be used to set up an on-premise private cloud behind organization’s firewall for gaining better control over infrastructure.

2.2 CloudStack Components

1. Host:A host is a computer that provides all computing resources such as CPU, storage, memory and networking to run the virtual machines.
2. Hypervisor:Hypervisor is software which will help us in implementing virtualization. KVM, VMWare ESX and Xen are some examples of Hypervisors.

IJARSE

3. Virtual machine: Virtual machine is a virtual hardware allocated by Hypervisor for installing Guest OS on it so that it can run as separate machine. Each host has a hypervisor installed to manage the VMs. As CloudStack is hypervisor agnostic, multiple hypervisor-enabled servers such as a Linux KVM-enabled server and a Citrix XenServer server can be used.
4. Cluster:One or more primary storage is coupled with a cluster that stores the disk volumes for all VMs running on hosts in that specific cluster. A cluster can be considered as a set of XenServer servers or a set of KVM servers.
5. Storage server:A cluster consists of one or more hosts and one or more primary storage servers.
6. Rack: CloudStack pod represents a single rack. A CloudStack pod consists of one or more clusters of hosts and one or more primary storage servers. Hosts in the same pod are in the same subnet.

III HYPERVISOR

A Hypervisor is an ideal delivery mechanism which acts as a traffic cop to make things happen in an orderly manner. The hypervisor is an operating system which resides at the lowest levels of the hardware environment because cloud computing support many different operating environments. Hypervisor shows the same application on lots of systems without having to physically copy that application onto each system. In other words, hypervisor is a practical way of getting things virtualized quickly and efficiently [7].

The term hypervisor was first introduced in 1956 by IBM with the hypervisor program installed on the computer allowed memory sharing. The hypervisor installed on the server hardware controls the guest operating system which is running on host machine. Hypervisor's main job is to cater to needs of guest operating system and effectively manage so that the instances of multiple operating systems do not interrupt one another. A virtual machine can create requests to the hypervisor through API calls method [7, 8]. There are two types of hypervisors: Embedded or hosted hypervisors, and Bare metal or native hypervisors which are discussed below:

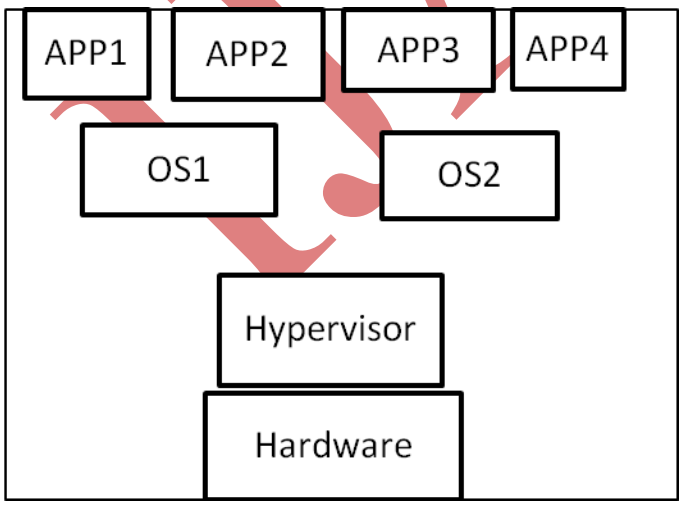

Figure 1: Type-1 Hypervisor

Type 1: Type-1 hypervisor is also known as native or bare-metal hypervisors which runs directly on the host computer's hardware to control the hardware resources and to manage guest operating systems. Examples of Type 1 hypervisors include VMware ESXi, Citrix XenServer and Microsoft Hyper-V hypervisor.

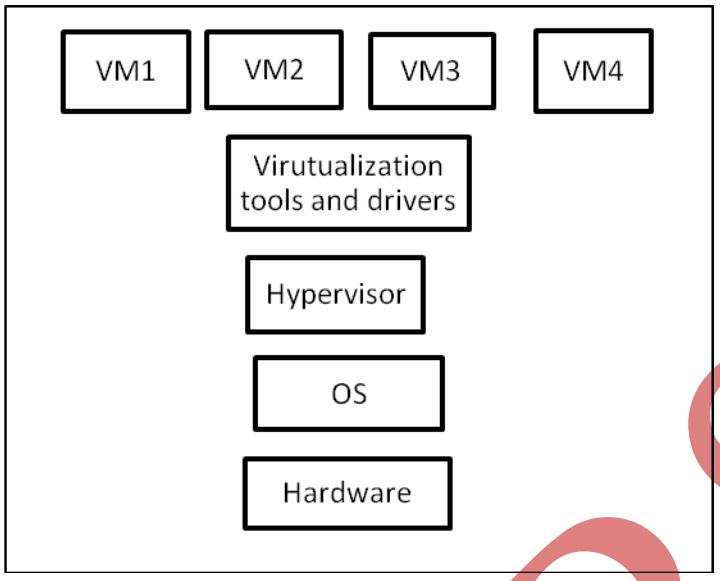

Figure 2: Type II Hypervisor

Type 2: Type 2 hypervisor is also known as hosted hypervisors which runs within a formal operating system environment. In this type, the hypervisor runs as a distinct second layer while the operating system runs as a third layer above the hardware.

The servers would need to execute the hypervisor. The hypervisor, in turn, loads the client operating systems of the virtual machines. The hypervisor allocates the correct CPU resources, memory, bandwidth and disk storage space for each virtual machine.

3.1 HyperThreading

Hyper-Threading is a technology used by some Intel microprocessors that allows a single microprocessor to act like two separate processors to the operating system and the application program s that use it. With Hyper-Threading, a microprocessor's core processor can execute two concurrent threads of instructions sent by the operating system. Having two threads of execution units to work on allows more work to be done by the processor during each clock cycle. To the operating system, the Hyper-Threading microprocessor appears to be two separate processors. Because most of today's operating systems are capable of dividing their work load among multiple processors, the operating system simply acts as though the Hyper-Threading processor is a pool of two processors [7].

IJARSE

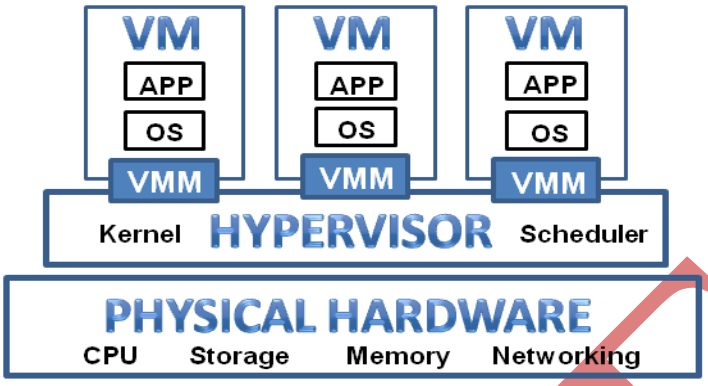

Figure 3: HyperThreading Overview

Hyper-threading is a technology which is used in certain Pentium 4 processors and all Intel Xeon processors. While hyper-threading can improve processing performance, software must support multiple processors to take advantage of the technology. Fortunately, recent versions of both Windows and Linux support multiple processors and therefore benefit from hyper-threading. Hyper-threading allows the two programs to be processed as separate threads at the same time. However, individual programs can only take advantage of Intel's HT Technology if they have been programmed to support multiple processors.

IV VIRTUAL MACHINE

From the technical point of view, two pillars of cloud computing are service-oriented architecture (SOA) and virtualization of hardware and software [1]. The important progress seen lately in virtualization is due to the development of several virtual machine hypervisors. Many years ago, a problem aroused. How to run multiple operating systems on the same machine at the same time? The solution to this problem was virtual machines. Virtual machines monitor the core part of virtual machines sits between one or more operating systems and the hardware and gives the illusion to each running OS that it controls the machine. Virtual Machine technology begins to emerge as a focus of research and deployment. Virtual Machine technology enables multiple OS environments to coexist on the same physical computer in strong isolation from each other. VMs share the conventional hardware in a secure manner with excellent resource management capacity, while each VM is hosting its own operating system and applications [4, 5].

Indeed, the VMM serves as an operating system for operating systems, but at a much lower level; the OS must still think it is interacting with the physical hardware. Thus, transparency is a major goal of VMMs [5]. Today virtual machine has become popular due to various reasons. Server consolidation is one such reason. In many settings, people run services on different machines which run different operating systems, and yet each machine is lightly utilized. In this case, virtualization enables an administrator to consolidate multiple Operating systems onto fewer hardware platforms, and thus lower costs and ease administration [5, 6].

IJARSE

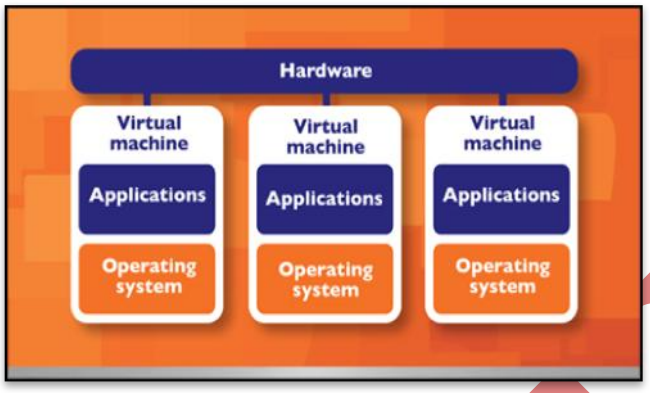

Figure 4: Overview of VM Architecture

4.1 KVM

KVM hypervisor is the virtualization layer in Kernel-based Virtual Machine (KVM). A hypervisor is a program that allows multiple operating systems to share a single hardware host. KVM is free, open source virtualization architecture for Linux distributions. KVM virtualization is often compared with Xen, which is the open source hypervisor for Oracle VM, Citrix Systems Inc.'s XenServer. But KVM virtualization, which is supported by Red Hat and Canonical uses a type-two hypervisor that resides within the Linux kernel. In KVM, the Linux kernel acts as a streamlining management and improving performance in virtualized environments. The hypervisor creates VM environments and coordinates calls for processor, memory, hard disk, network, and other resources through the host OS [7, 8] .

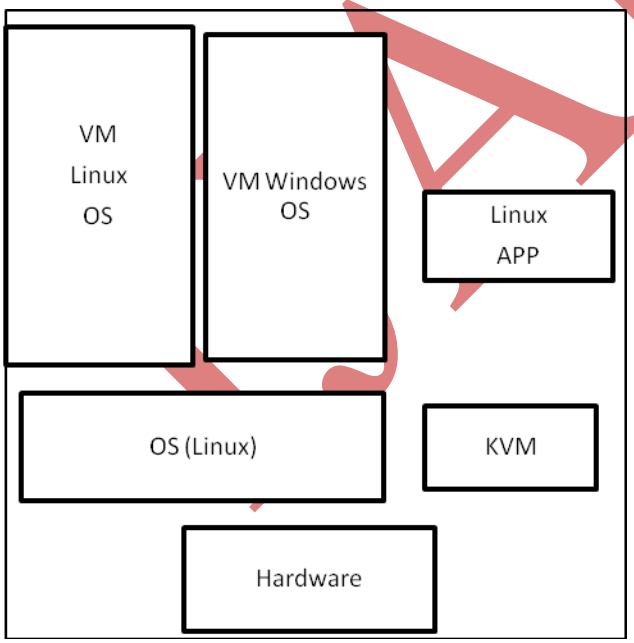

Figure 5: KVM Architecture

Numerous guest OSs can work with KVM including BSD (Berkeley Software Distribution), Solaris, Windows, Haiku, ReactOS, Plan 9, and the AROS Research OS. KVM is a full virtualization solution for Linux on x86

hardware containing virtualization extensions. It consists of a loadable kernel module, kvm.ko that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko. KVM also requires a modified QEMU although work is underway to get the required changes upstream. Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. The kernel component of KVM is included in mainline Linux, as of 2.6.20. There are many advantages of this KVM when compared to other virtualization software's available in Linux.

1. Can interact directly with the Kernel
2. Default virtualization in leading Linux Distributions
3. One of the Linux software developed aggressively.
4. Almost becoming competitor to VMware by implementing technologies such as v2v, p2v, and many open source tools to manage VM's
5. Number of open source cloud automation software's use KVM as default hypervisor.
6.

4.1.1 KVM Virtualization

KVM virtualization architecture is relatively new Because of KVM is embedded in the Linux kernel. KVM is easier to manage the all virtual machines and Linux updates. Once it is installing KVM on a Linux box a hardware file /dev/kvm is created which will act as interpreter between actual hardware and hypervisor manager. Whenever a request comes for hardware updating from hypervisor manager, the KVM software starts allocating those resources virtually by interacting with real hardware.

Suppose there is requirement to change RAM on a virtual machine, this is communicated by hypervisor manager to KVM for allocating the resource. Then KVM interacts with hardware and reserves that RAM from real RAM for that particular VM. This happens for the other resources as well.

If hardware supports virtualization directly without any third-party software to simulate then that hardware is called as Virtualization Technology enabled processor (VTx) in Intel processors and AMD-v For AMD processors.

4.1.2 KVM Architecture and Tools

In the KVM architecture the virtual machine is implemented as regular Linux process, schedule by the standard Linux scheduler. In fact each virtual CPU appears as a regular Linux process. This allows KVM to benefit from all the features of the Linux kernel. Device emulation is handling by a modified version of QEMU that provides emulated BIOS, PCI bus, USB bus and a standard set of devices such as IDE, SCSI disk controllers and network cards. There are broadly two KVM management tools:

1. Virsh: From the command line, virsh can streamline KVM management due to it is a master command with numerous subcommands, the learning curve is steep.
2. Virtmanager:Virt-manager is a graphical user interface that simplifies the management of virtual machines in Red Hat Enterprise Linux.

Virsh has a larger feature set, but virt-manager's point-and-click interface can perform most administrative tasks.

IJARSE

V BENCHMARK

To evaluate the performance of the Cloud infrastructure, I have selected a scientific computing application, a well known synthetic benchmark, the Linpack numerical library [7, 11]. Using this application, I will evaluate the impact in performance of several key factors in cloud computing: the number of virtual machines per host, the influence of the hyper threading and the hard disk I/O. The core of Linpack applications is the solution of linear systems of equations, which are dense matrices for Linpack. Next, I am going to describe the main features of each application.

5.1 LINPACK (LINear system PACKage)

The LINPACK package is a collection of FORTRAN subroutines for solving various systems of linear equations and linear least-squares problems. The software in LINPACK is based on a de-compositional approach to numerical linear algebra. The package has the capability of handling many different matrix and data types and provides a range of options. The package solves linear systems whose matrices are general, banded, symmetric indefinite, symmetric positive definite, triangular, and tridiagonal square. It also computes the QR and singular value decompositions of rectangular matrices and applies them to least-squares problems. LINPACK uses column-oriented algorithms to increase efficiency by preserving locality of reference. The LINPACK package was based on another package which is called Level 1 Basic Linear Algebra Subroutines (BLAS) library. BLAS carried out most of the floating-point work within the LINPACK algorithms. BLAS also work out with different linear equations and linear least-squares problems and makes it possible to take advantage of special computer hardware [5-8].

In the LINPACK Benchmark, a matrix of size 100 was originally used because of memory limitations with the computers. Such a matrix has 10,000 floating-point elements and could have been accommodated in most environments of that time [5, 6]. This was done so users could estimate the time required to solve their matrix problem by extrapolation. Over the years additional performance data was added and today the collection includes over 1300 different computer systems. In addition to the increasing number of computers, the scope of the benchmark has also expanded.

LINPACK was designed for applying to supercomputers in the early 1980s and now acts as one of the most authoritative benchmarks in high performance computers. The TOP 500 computers in the world are sorted by the LINPACK's result. To follow the development of computer architectures, LINPACK evolves into EISPACK and LAPACK. EISPACK mainly dedicates to numerical computation of the Eigen values and eigenvectors of matrices. LINPACK measures the actual peak value of float-point computing power indicated in giga of float-point operations per second (GFLOP) [7, 9].

IJARSE

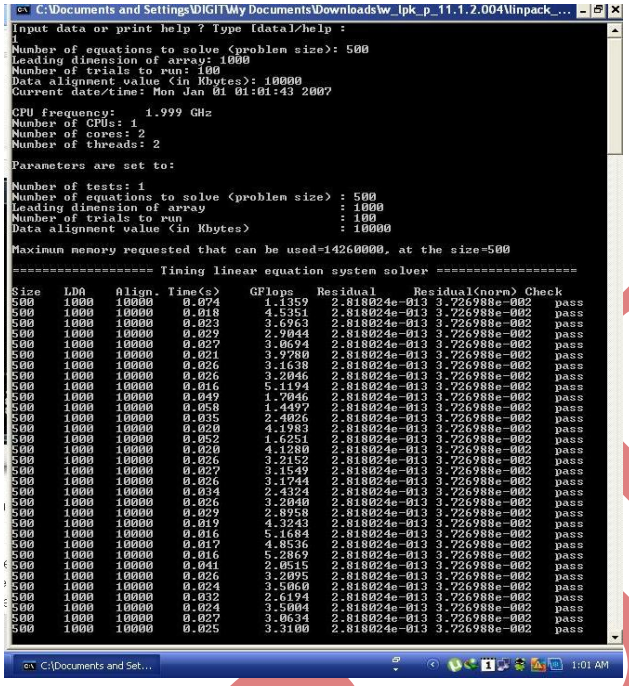

Figure 6: A sample output from Linpack 100

LINPACK was chosen because it is widely used and performance numbers are available for almost all relevant systems. A detailed description as well as a list of performance results on a wide variety of machines is available in postscript form from netlib. The benchmark used in the LINPACK Benchmark is to solve a dense system of linear equations. For the TOP500, I used that version of the benchmark that allows the user to scale the size of the problem and to optimize the software in order to achieve the best performance for a given machine. This performance does not reflect the overall performance of a given system, as no single number ever can. It does, however, reflect the performance of a dedicated system for solving a dense system of linear equations. Since the problem is very regular, the performance achieved is quite high, and the performance numbers give a good correction of peak performance.

VI CONCLUSION

Cloud technologies are drawing attention from industry, research centers and IT community due to they offer the possibility to virtualized services, providing virtual machines to users for their execution. Currently, there is a wide range of open-source solutions for building private, public or even hybrid Clouds. In this paper I have used the KVM hypervisor included in this platform to manage the virtualized services. The objective is to analyses the influence of the HyperThreading, the number of virtual machines employed per host and the I/O on the performance of LinPack parallel benchmark.

REFERENCES

[1] L. J. Zhang, J. Zhang, J. Fiaidhi and J. M. Chang, "Hot Topics in Cloud Computing", IEEE IT Professional, Vol. 12, No. 5, PP: 17-19, 2010.

[2] M. T. Jones, "Discover- Linux Kernel Virtual Machine", EMULEX Corp. IBM-Developer Works, 2007

[3] I. Habib, "Virtualization with KVM", Linux Journal, 2008

[4] F. Gomez-Folgar, J. L´opezCacheiro, C. Fern´andezS´anchez, A. Garcia-Loureiro and R. Valin, "An e-Science infrastructure for nanoelectronic simulations based on Grid and Cloud technologies", Spanish Conference Electron Devices (CDE), PP: 1-4, 8-11, 2011

[5] A.J. Garc´ıaLoureiro, T.F. Pena, J.M. L´opezGonz´alez and Ll. Prat, "Parallel implementation of a simulator for hetero junction bipolar transistors, VIII Symp. on Parallelism", C´aceres, pp. 41-50, 1997

[6] J. J. Dongarra, C. B. Moler,J. R. Bunch and G.W. Stewart, "LINPACK Users' Guide", SIAM publishing, 1979.

[7] Natalia Seoane, Raul Valin, Antonio J. Garcia-Loureiro, Tom´as F., Pena, "Performance of OpenMP simulations on the Cloud", Department of Electronics & Computation, University Santiago de Compostela, Supercomputing Center of Galicia (CESGA), 2013.

[8] S. Selberherr, "An Analysis and Simulation of Semiconductor Devices", Springer, 1984

[9] D.L. Scharfetter and H.K. Gummel, "Large-Signal Analysis of a Silicon Read Diode Oscillator, IEEE Trans. On Electron Devices, pp. 64-77, 1969

[10] R.E. Bank and D.J. Rose, "Parameter Selection for Newton-Like Methods Applicable to Nonlinear Partial Differential Equations", SIAM J. Num. Anal., Vol. 17, No. 6, pp. 806-822, 1980

[11] J.J. Dongarra, P. Luszczek and A. Petitet. "The LINPACK Benchmark: past, present and future", Concurrency and Computation Practice and Experience, vol: 15 issue: 9, PP: 803−820, 2003.

[12] Edward Anderson et. al., "LAPACK User's Guide", Society for Industrial and Applied Mathematics, Philadelphia, PA, Third edition, 1999.

[13] R. Venugopal, Z. Ren, S. Datta, M. S. Lundstrom, and D. Jovanovic, Simulating quantum transport in nanoscale transistors: Real versus mode–space approaches, Journal of Applied Physics, Vol. 92, No. 7, p. 3730, 2002

[14] Carlo Jacoboni and Paolo Lugli, The Monte Carlo Method for Semiconductor Device Simulation, Springer–Verlag Wien New York, 1989

[15] KVM, http://www.linux-kvm.org

[16] CloudStack, http://cloudstack.org/

[17] Linpack, http://software.intel.com

[18] Top 500 list, http://www.top500.org/project/linpack

[19] The virtualization API, http://libvirt.org

IJARSE