# Performance Evaluation of Xen, KVM, and Proxmox Hypervisors

**Article** *in* International Journal of Open Source Software and Processes · April 2018

DOI: 10.4018/IJOSSP.2018040103

Abdullah Algarni
Imam Muhammad bin Saud Islamic University
**2** PUBLICATIONS **17** CITATIONS
SEE PROFILE

Roobaea Alroobaea
Taif University
**86** PUBLICATIONS **378** CITATIONS
SEE PROFILE

Ahmed Ghiduk
Georgia Institute of Technology
**28** PUBLICATIONS **337** CITATIONS
SEE PROFILE

Farrukh Nadeem
King Abdulaziz University
**36** PUBLICATIONS **548** CITATIONS
SEE PROFILE

International Journal of Open Source Software and Processes
Volume 9 • Issue 2 • April-June 2018

# Performance Evaluation of Xen, KVM, and Proxmox Hypervisors

Sultan Abdullah Algarni, Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudia Arabia

Mohammad Rafi Ikbal, King Abdulaziz University, Jeddah, Saudia Arabia

Roobaea Alroobaea, College of Computers and Information Technology, Taif University, Ta'if, Saudi Arabia

Ahmed S Ghiduk, College of Computers and Information Technology, Taif University, Ta'if, Saudi Arabia

Farrukh Nadeem, Department of Information Systems, King Abdulaziz University, Jeddah, Saudia Arabia

## ABSTRACT

Hardware virtualization plays a major role in IT infrastructure optimization in private data centers and public cloud platforms. Though there are many advancements in CPU architecture and hypervisors recently, but overhead still exists as there is a virtualization layer between the guest operating system and physical hardware. This is particularly when multiple virtual guests are competing for resources on the same physical hardware. Understanding performance of a virtualization layer is crucial as this would have a major impact on entire IT infrastructure. This article has performed an extensive study on comparing the performance of three hypervisors KVM, Xen, and Proxmox VE. The experiments showed that KVM delivers the best performance on most of the selected parameters. Xen excels in file system performance and application performance. Though Proxmox has delivered the best performance in only the sub-category of CPU throughput. This article suggests best-suited hypervisors for targeted applications.

## KEYWORDS

Cloud Computing, Hypervisor, Performance, Virtualization

## 1. INTRODUCTION

The advent of hardware virtualization technology has laid the foundation for many advanced technologies, such as cloud computing, IT infrastructure optimization and consolidation, disaster recovery, high availability and green computing. Hardware virtualization allows many guest operating systems to share the same hardware, as shown in Figure 1. This is done by installing a hypervisor on physical hardware and then installing guests on top of the hypervisor. The hypervisor manages all the physical resources of the host system, like CPU, memory, network and storage.

Hypervisors allow not only multiple guest operating systems to share the same physical hardware, but also allow the sharing of abstract physical hardware such that guest operating systems assume that they are running on physical hardware. This abstraction has many advantages. Hypervisors simplify resource management, speed up deployment, use resources more efficiently and offer better control over infrastructure.

Hypervisors can create pseudo hardware resources for guests that are idle and use these resources for guests that are loaded and in need of resources. This also helps with many advanced features like

DOI: 10.4018/IJOSSP.2018040103

thin provisioning, virtual machine migration and high availability. However, the disadvantage of the presence of this abstraction prevent it from being used in mission-critical and performance-demanding applications like high-performance computing,

In cloud computing, both private and public clouds leverage features of hypervisors to deliver infrastructure as a service (IaaS) to meet end user demands like instant operating system deployment, storage allocation, network management and configuration. These cloud infrastructures are easily scalable and flexible as virtual servers can be created and customized in almost no time. IT infrastructure consolidation is also an area where virtualization is implemented. One single physical server is used to deploy multiple diverse operating systems as per end-user requirements, maximizing resource utilization and reducing operating costs for power, cooling and rack space.

The success of virtualized IT infrastructure depends on physical hardware and hypervisors. There is continuous development in chip technologies, thus server hardware is getting better over time. This paper performs a comprehensive performance evaluation and benchmark three main open source hardware-assisted hypervisors: XenServer (2017), KVM (Kernel Virtual Machine) (Linux, 2017) and Proxmox VE (2017) on different areas, such as response efficiency, CPU cache & throughput, performance of Memory & Disk & Application. The aim of this research is to enable researchers and end users to determine which hypervisors deliver the best performance for the targeted applications to meet their needs based on hands on experiment.

The remainder of this work is structured as follows. The following section presents the related work. Section 3 demonstrates a short explanation of the selected hypervisors Xen, KVM and Proxmox. Following that, section 4 shows the details of the selected parameters and benchmarks in the experiment, the setup of the experiment and the results of the evaluation of the three hypervisors. Section 5 presents the conclusion of this paper and some future work issues.

## 2. RELATED WORK

There are countless studies evaluating the performance of hypervisors from various perspectives. Many authors focus on analyzing the complete cloud computing which uses hypervisor such as Nadeem and Qaiser (2015), Paradowski, Liu, and Yuan (2014), Al-mukhtar (2014), Younge et al. (2011),

**Figure 1. Typical virtualization architecture**

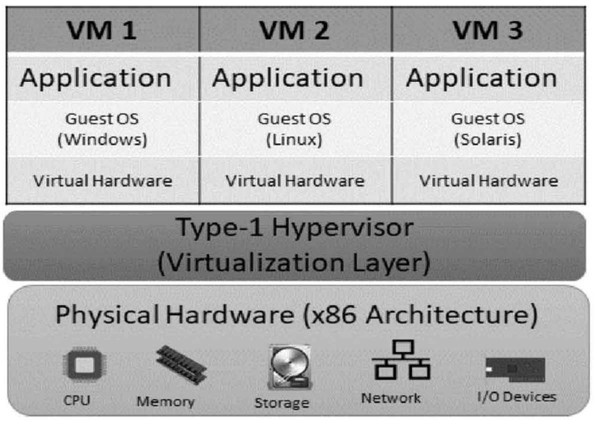


**Fig. 1.** Typical virtualization architecture

and Graniszewski and Arciszewski (2016). Paradowski et al. (2014) used a mutual hypervisor under defined criteria. This study used various resources performances such as CPU, RAM and hard disk size. The results indicated that OpenStack outperformed CloudStack based on the benchmark criteria.

A few studies investigate the performance of virtualization methodologies for high performance computing such as Younge et al. (2011). This study suggested that KVM is the best option to be utilized with high performance computing cloud environments.

However, our study focuses on analysing hypervisors only. Many researchers compare Xen and KVM, using different benchmarks such as Deshane et al. (2008), Andrea Chierici et al. (2010), Geoffrey et al. (2016), Pagare et al. (2014) and Binu and Kumar (2011). They all evaluate the performance of Xen and KVM, concentrating on overall performance and individual components, such as scalability of the virtual machines, network, memory, disk I/O and CPU performance, using benchmarking tools such as bonnie++, SysBench, IOzone, netperf, iperf and hep-spec06. They have concluded that there is no perfect hypervisor for all kinds of environments and task allocations, and both hypervisors handle perfectly the workloads that are best suited for them.

Binu and Kumar (2011) compared Xen and KVM hypervisors. The authors illustrated that scalability was the most obvious difference. Moreover, guests crashing was noticed with KVM once the number of guests exceeded four. Though, KVM outperformed Xen in isolation and I/O-intensive tests, whereas Xen outperformed KVM in a kernel compile test.

Hwang et al. (2013) presented a competitive study of the performance of Hyper-V, KVM, vSphere and Xen hypervisors. Manik and Arora (2016) compared KVM, Xen, Hyper-V and ESXi. Both studies indicated that there is no perfect hypervisor that always outperforms the others in all aspects, as different workloads may be best suited for different hypervisors.

Elsayed and Abdelbaki (2013) conducted a qualitative and quantitative performance assessment for VMware ESXi5, Microsoft Hyper-V2008R2, and Citrix Xen Server 6.0.2. The benchmarks used to measure the performance were the DS2 test application and the PRTG network monitor tools. They focused on DB performance and used customized SQL instances.

Graniszewski and Arciszewski (2016) compared Hyper-V, ESXi, OVM, VirtualBox, and Xen. They used CPU (nbench), NIC (netperf), storage (Filebench), memory (ramspeed) compare efficiency the target hypervisors. Their results showed that ESXi is the best hypervisor in VM performance.

In contrast to previous work, this study is different in scope and to the best of knowledge there is any work that compares the performance of KVM, Xen and Proxmox in a comprehensive matter. This paper compared the performance of the three hypervisors in an environment configured mainly for performance evaluation, using the up-to-date versions of the hypervisors to ensure the right decision about selecting one over the other.

## 3. HYPERVISORS

### 3.1. Xen

This section describes the three hypervisors studied (Xen, KVM and Proxmox). The selected hypervisors are popular open-source hypervisors embraced by public and private cloud environments. Private cloud platforms like Openstack, Cloudstack and OpenNebula (2017) support Xen and KVM hypervisors. These hypervisors are also preferred for standalone implementations in data centres.

Xen (XenServer Open Source Server Virtualization) version 4.8.1 is a type 1 hypervisor developed by the University of Cambridge computer laboratory. It is currently an open-source, community-driven hypervisor. It is a sophisticated server virtualization platform which is used to manage mixed operating system. Xen supports both full virtualization and para virtualization; this paper has used full virtualization. Notable public cloud platforms like Amazon EC2 (2017) and Rackspace (2017) use Xen. Xen is best known for its performance and scalability under heavy workloads. These features make Xen a preferred open-source hypervisor for many administrators.

Figure 2. KVM architecture

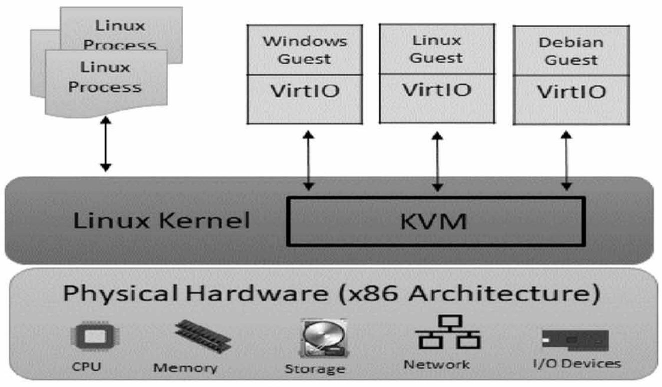


**Fig. 2.** KVM architecture

Since Xen is a type 1 hypervisor, it is the first program that runs directly on the top of system hardware. The guest VMs run on top of the Xen hypervisor, these guest operating systems are called ad domains. Xen allows diverse operating system to run on top of it and provides completely isolated resources for its guests as per configuration. Xen also allows individual guest operating systems to run their own versions of device drivers for easy and isolated management.

## 3.2. KVM

KVM (Kernel Virtual Machine) (2017) is a full virtualization developed by Qumranet and later acquired by Red Hat (2017) in 2008. It is released under a GPL license. KVM is used in public cloud platforms like Google (Saavedra & Smith, 1996). KVM requires hardware virtualization extensions of the CPU like Intel VT or AMD-V. The kernel module KVM is designed for hosting the kernel of Linux and turns a standard Linux distribution into a hypervisor, in our experiments we have deployed KVM on CentOS 7.2. Scheduling the processes of the guest and memory is done by the host kernel. Hardware emulation is handled by QEMU. The host kernel considers the guest operating systems as regular processes. Figure 2 shows the architecture of KVM.

KVM does not perform any emulation by itself; it augments the traditional kernel by exposing / dev/kvm in guest mode. Each process is executed on the guest CPU of KVM which is implemented by a Linux thread. Using this feature, one can leverage normal Linux commands and utilities to set priorities and affinities to the guest processes. Control groups can also be implemented to control resource allocation to each guest VM running on the host. Guest operating systems' memory address space is allocated by the host kernel scheduler. Therefore, it can be shared, swapped or backed up by host's virtual memory and is also NUMA-aware. KVM supports device I/O using virtio. Virtio is a standard for device drivers for physical devices, like storage and network. In Virtio a guest's device driver is aware of the presence of the virtual environment. Virtio communicates directly with the hypervisor so that it gets high-performance access to the host's physical hardware devices.

## 3.3. Proxmox VE

The third open-source virtualization platform is Proxmox Virtual Environment (2017) version 5.0. Proxmox depends on Debian-based Linux distribution and uses web GUI for management. Proxmox is a full-virtualization hypervisor supporting virtual machines and Linux containers (LXC). Proxmox

Table 1. Description of Benchmarks

| Benchmark | Parameter |
|---|---|
| Sqlite | Application performance |
| LZMA | Application performance |
| Apache | Response efficiency |
| CacheBench | Cache performance |
| John the Ripper | CPU throughput |
| IOZone | Disk performance |
| RAMspeed | Memory performance |

is one of the few virtualization platforms supporting both full virtualization and Linux containers. Proxmox has many powerful features like live migration, high availability, bridged networking, flexible storage, deployment using templates, backup and Linux command line tools. Proxmox internally uses KVM for full virtualization and LXC for Linux containers. Proxmox can be clustered using multiple physical hosts to use features like live migration of guest VMs during maintenance and high availability. Proxmox uses pmxcfs, its own clustered file system, to implement cluster features. Proxmox has pre-packaged, commonly-used server appliances which are possible to download and deploy instantly, avoiding the need for installation and configuration of commonly used applications.

## 4. PERFORMANCE EVALUATION

This section will describe our performance evaluation model, benchmarks selected for the evaluation and detailed descriptions of each benchmark. Furthermore, our findings from our experiments will be analyzed.

### 4.1. Performance Assessment Model

The proposed model for the benchmarking study of performance is based on overall performance of the system, as well as performance of individual components using standard benchmarking methods. The proposed model starts by benchmarking the performance of Ubuntu Linux 16.04 (Xenial Xerus) on a standard server hardware and then benchmarking the same operating system installed as a guest on different hypervisors using the same physical hardware. The proposed model used traditional benchmarking systems to measure system performance and individual components like CPU, memory, cache and disk performance (Nadeem & Qaiser, 2015). Descriptions of each benchmark and the targeted systems are summarized in Table 1.

The intention for selecting these benchmarks is to drill down each resource component of each system, one by one, with a specific benchmark. The components include CPU, memory and disk I/O. Since private datacentres have dedicated high-speed networks and the performance of networks in real time depends on many parameters, benchmarking network performance is out of the scope of this study. Well-known benchmarks have been selected from different sources using diverse algorithms so that the results are unbiased, reliable and reproducible. In general, applications can be broadly classified into CPU, memory and disk I/O. With the results of this study, end-users and administrators will be able to compare their applications with the results of this paper to select a hypervisor best suited for their applications. Performance evaluation of individual components provides an estimate of overall performance of the system (Saavedra & Smith, 1996).

## 4.2. Benchmarks

This section describes the benchmarks and the corresponding parameters considered in this paper.

### *4.2.1. Response Efficiency*

This is a measure of time taken for the system to respond to a request from an application or client. This parameter is crucial for many systems, including authentication systems, database systems, web applications and file servers. It is measured in number of static web page requests a server or service can fulfil in one second. The Apache benchmark (2017) is used, which measures the number of static web page requests an Apache web server can serve.

### *4.2.2. CPU Throughput*

The CPU is one of the most important components of a system. This parameter is measured in computational work done per unit time. This benchmark is also an evaluation of latency in communicating process from guests to hosts and hypervisors' CPU scheduling policies. In this benchmark, the efficiency of the process management of the hypervisors is evaluated. Process management is a very important sub-system especially for CPU-intensive applications. This paper uses John the Ripper (OpenWall, 2017) to evaluate CPU performance. This benchmark is a common decipher application which uses diverse ciphers such as DES and MD5.

### *4.2.3. Memory Performance*

Memory management is one of the most important components of a system. A hypervisor employs different memory management techniques to provide memory to guest operating systems. At times, VMs can over-subscribe available physical memory. This process is known as memory overcommit. For example, there may be two VMs on a physical server serving users from two different time zones. However, these techniques may have adverse effects on the performance of VMs. During memory operations, RAMspeed (Alasir, 2017) is used to test memory bandwidth. This benchmark measures memory bandwidth of read, write and update actions. RAMspeed uses large blocks of data to carry out numerical operations of copy, scale, add and triad operation.

### *4.2.4. Cache Performance*

There is a significant delay for a CPU to access data from main memory because of the different speeds at which CPU and memory operate. Modern CPUs are equipped with caches at multiple levels to accelerate data transfers between main memory and the CPU. This mechanism is helpful, especially when CPU tries to access the same copy of data frequently. To evaluate the performance of the CPU cache at multiple levels CacheBench (University of Tennessee ICL, 2017) is used. The aim of using CacheBench is to deliver peak performance given optimum cache use. This benchmark performs different operations in the cache, like read, write and modify. Multiple iterations of these operations are performed with varying lengths of data over time, and cache bandwidth (in Mb per second) is computed by the division of the overall amount of data in megabytes by overall time.

### *4.2.5. I/O Performance of File System*

Almost all applications use storage sub-systems to store data permanently for later retrieval. Some applications read data in the beginning of processing and write results to storage while others read and write data as they process it. In either case, the performance of the storage sub-system is crucial for applications. In a normal system, a file system access from the operating system is passed to raid controllers, which fulfils a request directly. With the introduction of a hypervisor, file access requests from a guest operating system are sent to the hypervisor, which redirects the request to the storage controller. There is a slight performance degradation in this process with the introduction of the virtualization layer. IOzone benchmark is used to evaluate performance of file system. IOzone generates and assesses many file operations such as read, write and random read.

#### *4.2.6. Overall Performance*

Measuring the overall performance of a system is crucial, as the performance of many end-user applications depends on the performance of all the components of the system. Here, overall performance of the system is evaluated using two benchmarks. One is a compression benchmark and the other is a database application benchmark. For the file compression benchmark, LZMA (Lempel−Ziv−Markov Chain Algorithm) is used and for the database application, Sqlite is used.

LZMA is a lossless data compression technique with a great compression percentage. To evaluate the performance of the Sqlite database application, the number of transactions per second (TPS) is measured. TPS is the number of atomic operations a database performs in one second. The Sqlite application benchmark performs 12,500 insert operations in one second. TPS is calculated as the overall time of insert operations over 12,500.

## 4.3. Experimental Setup

The experiment is conducted on a Fujitsu Primergy RX2540 M2 server with a 2 GHz Intel Xeon E5-2660 v4 Processor, 32 GB memory and 1.6 TB of storage configured in RAID 0. Ubuntu Linux 16.04 was used as a base operating system to run benchmarks. For a fair comparison, Ubuntu was installed on bare metal, and benchmark results were taken. Then Ubuntu was installed as a guest operating system on top of each hypervisor with 2 CPU cores, 2GB memory and 20 GB storage. Benchmarks were executed on Ubuntu guests and results were taken. Benchmarks were executed three times and averages of the results were considered for comparison. We have rebooted the standalone, guest and host systems after each iteration of benchmark suite to make sure that the results are consistent. Though Ubuntu 16.04 deployed on bare metal has access to all the hardware resources (14 CPU cores, 32Gb RAM and 1.6 Tb of storage), the results are still comparable to performance of Virtual machines. CPU benchmark we used is serial in nature, Physical and Virtual machines can use only one CPU core for these tests. Memory benchmark tests the read-write efficiency of the memory and I/O benchmark tests read-write performance of disk with file size ranging from 2GB to 8GB only. Similarly, response efficiency and cache performance tests should not differ since we have deployed VMs with sufficient resources.

To make sure that our experiments are consistent and reproducible, the Phoronix test suite (201) was used to run benchmarks. The Phoronix test suite makes the process of running tests automatically which is extremely simple and reproducible. Qualitative and Quantitative benchmarks can be carried out by the test suite with statistically accurate results.

## 4.4. Results of the Experiments

This section discusses the results of the evaluation of the three open-source hypervisors. For a fair comparison, the benchmark suit is repeated several times and the average values of the results were taken into considerations in most cases to ensure correct and reliable values.

#### *4.4.1. Response Efficiency*

In this paper, response efficiency is measured as the number of static web pages an Apache web server can sustain when performing 1,000,000 requests, with 100 requests being executed concurrently. The greater the number of requests the system can sustain, the more effective the system. Figure 3 illustrations the number of requests sustained by the Apache web server on three hypervisors and on bare metal deployment. Since the system installed on bare metal has direct access to system resources, it has outperformed all the hypervisors in this benchmark. This paper will focus on hypervisors.

Applications running on Xen and Proxmox handled around 14,500 requests, while KVM handled 16,451 requests in one second. This is a performance increase of around 12%. In this category, KVM has outperformed Xen and Proxmox hypervisors.

**Figure 3. Response efficiency (number of requests per second)**

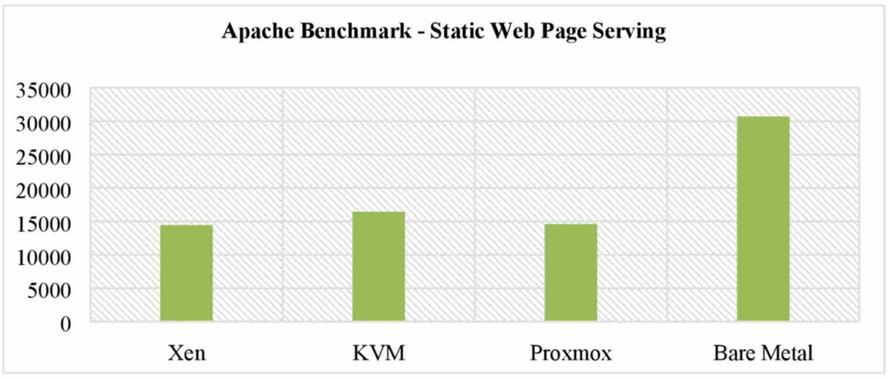


**Fig. 3.** Response efficiency (number of requests per second)

### *4.4.2. CPU Throughput*

John the Ripper benchmark is used to measure CPU throughput. This benchmark measures the number of username/password combinations handled by the CPU in one second. The benchmark is executed on three different ciphers: DES, MD5 and Blowfish. Figures 4(a)-4(c) show the number of ciphers the CPU processed per second for each of the hypervisors. KVM processed the highest number of username/password combinations for MD5 and Blowfish ciphers. Proxmox processed the highest number of traditional DES ciphers. KVM produces 5% and 2% higher performance than Xen and Proxmox, respectively, in processing the MD5 cipher. It also produced around 4% increased performance over the other two hypervisors in processing the Blowfish cipher. However, Proxmox delivered better performance in processing traditional the DES cipher by 1% and 5% over Xen and KVM, respectively. KVM has delivered the best overall performance in this category.

### *4.4.3. Memory Performance*

RAMspeed has been used to measure the memory performance of the system. This benchmark measures the performance of the system while completing different memory operations. The higher the output, the better the performance of the system. The results are recorded for both integer and float tests. A scientific application's performance depends on the number of floating point operations a system can perform in one second. Measuring the number of integer operations, the system can deliver is also an important metric to help determine the overall memory performance of a system. Figure 5 shows the memory performance which was achieved with the selected hypervisors and physical hardware in terms of memory bandwidth (MB/s) for diverse memory operations like copy, scale and add. Figures 5(a) and 5(b) show results of integer and floating-point operations, respectively. In both categories, KVM outperformed the other two hypervisors and physical hardware by 10.02% and 6.91%, respectively. In integer operation, Xen delivered the least performance after Proxmox by a difference of 137 MB/s. In floating point operations, Proxmox delivered the least performance after Xen by a difference of 234 MB/s.

**Figure 4. (a). CPU throughput (DES cipher)**

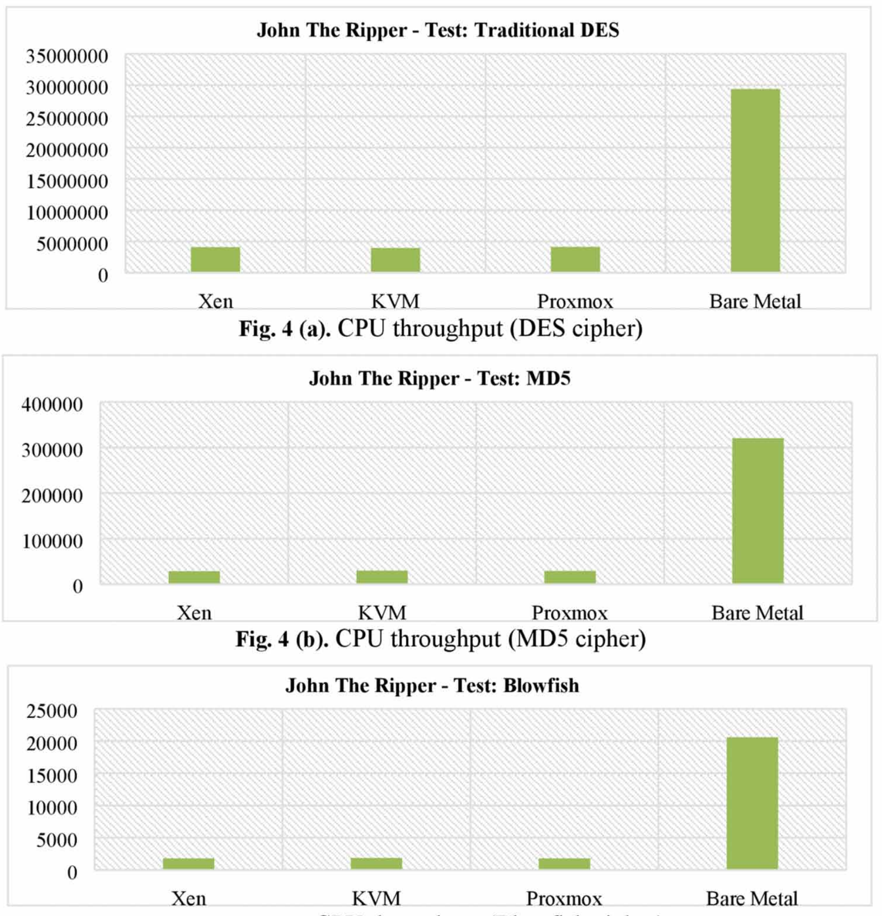


**Fig. 4 (a).** CPU throughput (DES cipher)

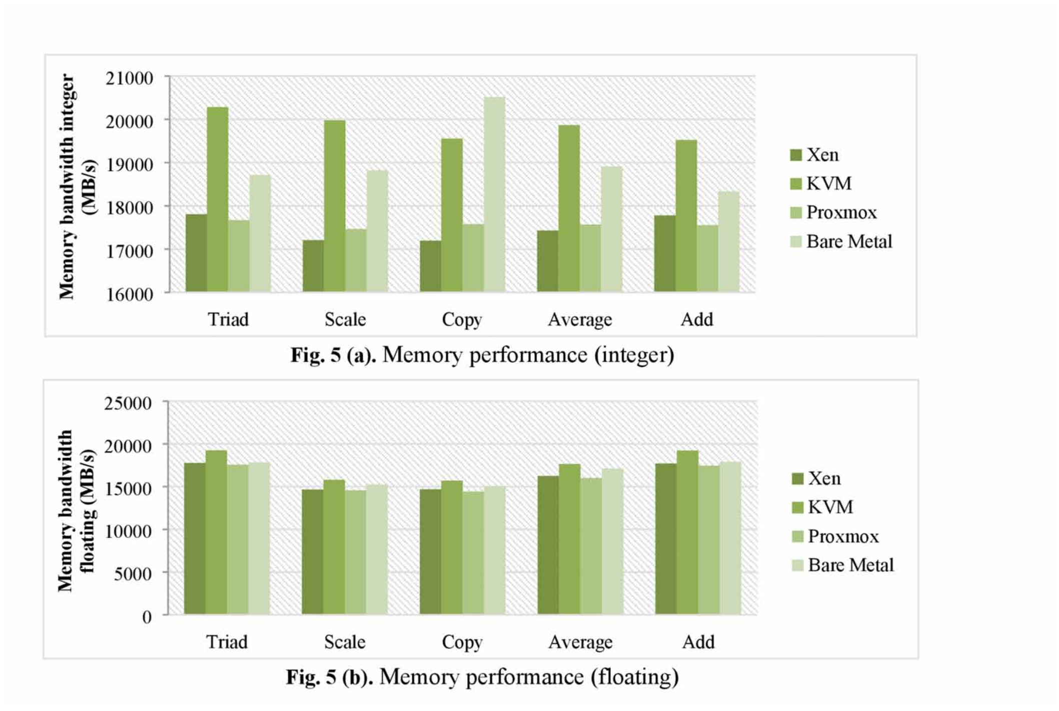
**Fig. 4 (b).** CPU throughput (MD5 cipher)

**Fig. 4 (c).** CPU throughput (Blowfish cipher)

### *4.4.4. Cache Performance*

Cache performance of the hypervisors and physical hardware was measured using the CacheBench benchmark. This benchmark measures performance in three categories: cache read, cache write, and cache modify. This is measured as the ability of the cache present in the system to sustain large, unit-stride, floating-point cache-optimized workloads. This is measured in raw bandwidth as megabytes per second, and the higher this metric is, the better the performance of the system. Figures 6(a)-6(c) show the results of the CacheBench benchmark executed on the three hypervisors and on the physical hardware. The physical hardware has again outperformed the three hypervisors. Out of the three hypervisors, KVM shows the highest cache performance and Xen shows lowest. The performance difference between KVM and Xen is approximately 2.5% in all three categories. Proxmox has delivered good performance after KVM. The performance difference between KVM and Proxmox is approximately 0.6% in all three categories.

Figure 5. (a). Memory performance (integer)



### 4.4.5. File System I/O Performance

IOzone benchmark is used to evaluate the performance of the storage sub-systems of the three hypervisors. IOzone evaluates the performance of storage sub-systems in two categories: disk read and disk write. Disk read and disk write provide the bandwidth for reading and writing files on the disk with varying record sizes. This is expressed in megabytes per second, and the higher the value the better the performance of the system. The experiments were conducted on a file size of 2GB with three records of different size of 4 KB, 64 KB and 1MB.

Figure 7(a) depicts the disk write performance of the three hypervisors and the physical hardware with the three record sizes. Xen delivered the best performance of all three hypervisors. Interestingly, in this case, Xen performed better than the physical hardware as well. KVM shows the lowest performance of the three hypervisors, performing at 57.46% less than Xen. Performance of Proxmox is almost near Xen with a variation of only 4%. The performance of the physical machine installed without any hypervisor performed at 32% less than Xen. If disk read performance from Figure 7(b) is analysed, one can see that the physical machine delivered better performance compared to the three hypervisors by around 70% in all three record sizes. Out of the three hypervisors, Xen delivered the best performance after the physical machine. Proxmox delivered the next best performance with a variation of around 6%. KVM delivered the least impressive performance and was less than Xen by approximately 17% in all three categories. From Figures 7(a) and 7(b), one can see that there is no major difference in performance across different record sizes. However, one can observe that disk-read operations are much faster than disk-write operations.

### 4.4.6. Application Performance

Two benchmarks were used to compare the performance of applications on the selected hypervisors and physical hardware. The LZMA benchmark measures the amount of time consumed in compressing a 256MB file using the LZMA compression algorithm. The lower the time, the better the performance

**Figure 6. (a). Cache performance (read operation)**

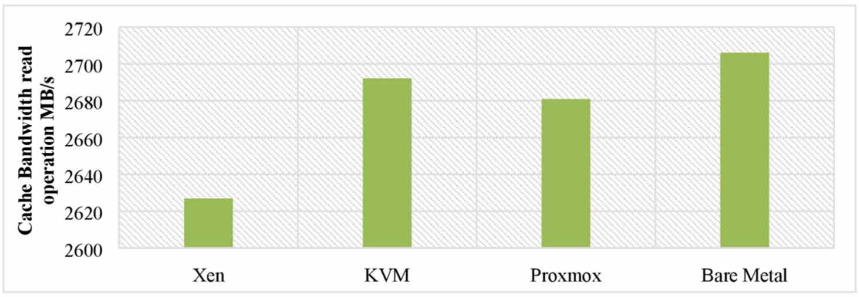


**Fig. 6 (a).** Cache performance (read operation)

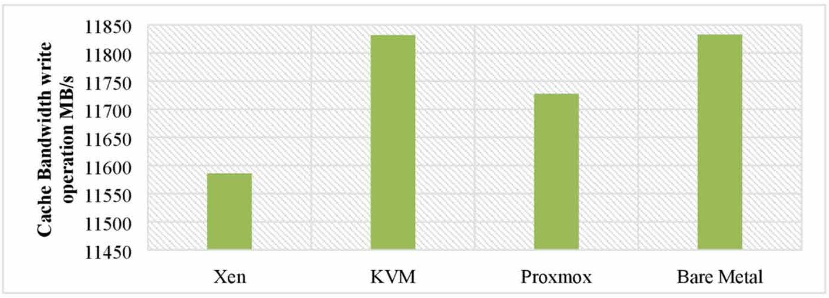


**Fig. 6 (b).** Cache Performance (write operation)

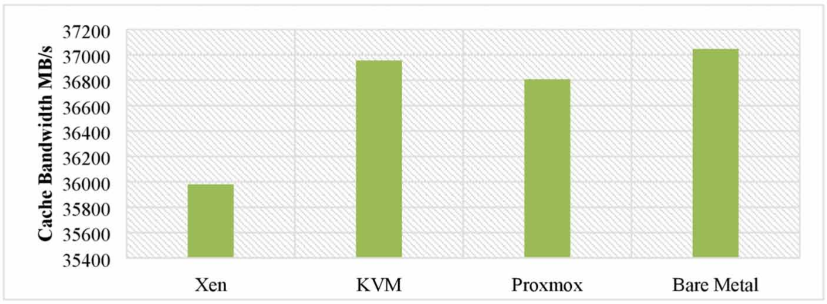


**Fig. 6 (c).** Cache performance (read/write/modify operation)

of the system. Figure 8(a) depicts the results. Of the hypervisors, KVM delivered the best performance, followed by Proxmox. Xen delivered the lowest performance of the three hypervisors with the same workload. The performance of KVM and Proxmox is very close with a difference of only 1.2%. The performance difference between KVM and Xen is 4.63%. KVM delivered the best performance in this benchmark test after the physical hardware.

Figure 8(b) shows the performance of the Sqlite benchmark. This benchmark measures the time taken to perform a pre-defined number of insertions on an indexed database. The performance of bare metal was 25% higher than best-performing hypervisor and 125% higher than the average performance of all three hypervisors. Xen has delivered the best performance of all three hypervisors, followed by KVM. There is a huge performance difference between the three hypervisors. Xen outperformed KVM with a difference of 82.86%. Proxmox delivered the least performance of the three hypervisors at 86% less than KVM and 143% less than the highest-performing Xen hypervisor.

Figure 7. (a). File system performance (disk write)

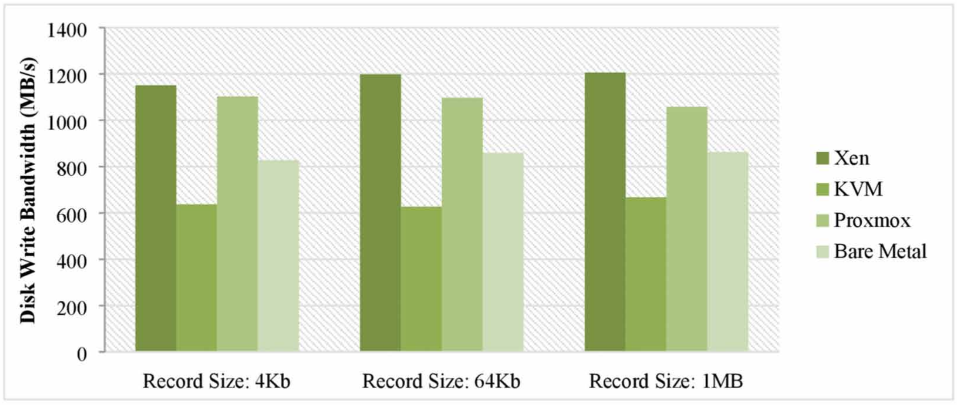


**Fig. 7 (a).** File system performance (disk write)

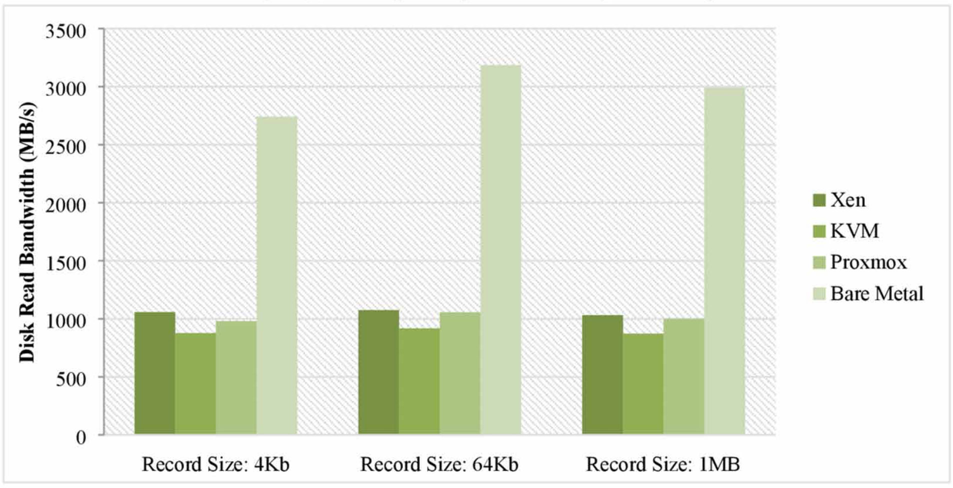


**Fig. 7 (b).** File system performance (disk read)

To conclude Table 1 summarizes the performance of the three hypervisors during our tests. From the selected parameters, KVM delivers the best performance on CPU throughput, response efficiency, memory performance and cache performance. Xen excels in file system performance and application performance. Though Proxmox has delivered the best performance in only the sub-category of CPU throughput, it has delivered consistent performance in all other categories.

## 5. CONCLUSION

This study presented our performance comparison analysis of KVM, Xen and Proxmox hypervisors. Also, the performance has been presented of physical hardware for comparison. Extensive experiments were conducted to evaluate their performance by installing Ubuntu guests on top of different hypervisors and on bare metal. We have included performance of bare metal deployment so that user can treat these results as base line for each benchmark category. This will also help reader to visualize the latency introduced by different hypervisor on comparison with performance of bare metal.

Figure 8. (a). Application performance (LZMA)

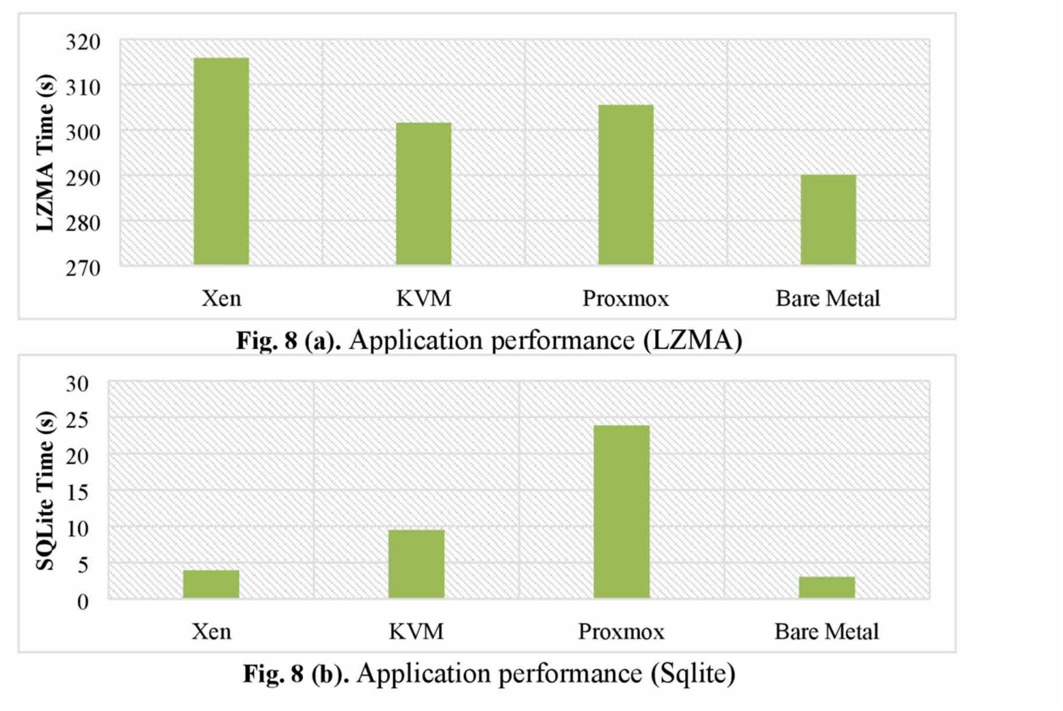


Table 2. Performance comparison of three hypervisors

| # | Performance Parameter | Benchmark | Benchmark Operation | Xen | KVM | Proxmox |
|---|---|---|---|---|---|---|
| 1 | Response Efficiency | Apache | Serving static pages | *LB | **BS*** | |
| 2 | CPU Throughput | John the Ripper | Blowfish | LB | **BS** | |
| | | | Traditional DES | | LB | **BS** |
| | | | MD5 | LB | **BS** | |
| 3 | Memory | RAMspeed | Integer (Average) | LB | **BS** | |
| | | | Float (Average) | | **BS** | LB |
| 4 | Cache Performance | CacheBench | Read | LB | **BS** | |
| | | | Write | LB | **BS** | |
| | | | Read/modify/write | LB | **BS** | |
| 5 | File System Performance | IOzone | Disk read | **BS** | LB | |
| | | | Disk write | **BS** | LB | |
| 6 | Application Performance | LZMA | File compression | **BS** | LB | |
| | | Sqlite | Database inserts | **BS** | | LB |

* LB represents Least Performance and BS represents Best Performance

From our experiments we see that KVM has outperformed in most of the categories except File system performance, while Xen has outperformed only in File system performance. This shows that KVM is best suited for application which are CPU and memory intensive like web applications and Xen is best suited for applications which depends on File system or storage. Database and storage-based applications will perform better if deployed on Xen hypervisor.

## 5.1. Future Work

There are few benchmarks where the performance of a virtual machine exceeds the performance of physical hardware. Depth analysis to understand these benchmarks will be performed in the future.

LXC (Linux Containers) is an operating system level virtualization technology that run multiple diverse operating systems on a single Linux kernel. The performance of Linux containers and hardware-assisted virtualization technologies will be compared in the future. Besides, this study will be extended to understand the performance of different virtualization technologies on cloud platforms.

## REFERENCES

AL-Mukhtar, M. M., & Mardan, A. A. A. (2014). Performance evaluation of private clouds Eucalyptus versus CloudStack. International Journal of Advanced Computer Science and Applications, 5(5), 108-117.

Alasir. (2017, June). RAMspeed, a cache and memory benchmark. Retrieved from http://alasir.com/software/ramspeed/

Amazon. (2017, June). Elastic compute cloud (EC2): Cloud server and hosting. Retrieved from https://aws.amazon.com/ec2/

Apache. (2017, June). Apache CloudStack: Open source cloud computing. Retrieved from https://cloudstack.apache.org/

Binu, A., & Kumar, G. S. (2011). *Virtualization techniques: A methodical review of XEN and KVM* In Advances in Computing and Communications (pp. 399–410).

Chierici, A., & Veraldi, R. (2010). A quantitative comparison between xen and kvm. *Journal of Physics: Conference Series*, *219*, 42005. doi:10.1088/1742-6596/219/4/042005

Deshane, T., Shepherd, Z., Matthews, J., Ben-Yehuda, M., Shah, A., & Rao, B. (2008). Quantitative comparison of Xen and KVM. In Xen Summit, Boston, MA.

Elsayed, A., & Abdelbaki, N. (2013). Performance evaluation and comparison of the top market virtualization hypervisors. In *2013 IEEE Int. Conf. on Computer Engineering and Systems*, . doi:10.1109/ICCES.2013.6707169

Graniszewski, W., & Arciszewski, A. (2016). Performance analysis of selected hypervisors (Virtual Machine Monitors - VMMs). *International Journal of Electronics and Telecommunications.*, *62*(3), 231–236. doi:10.1515/eletel-2016-0031

Hwang, J., Zeng, S., & Wood, T. (2013). A component-based performance comparison of four hypervisors. In *2013 IFIP/IEEE Int. Symp. On Integrated Network Management*. IEEE.

Linux. (2017, June). KVM. Retrieved from https://www.linuxkvm.org/page/Main_Page

Manik, V. K., & Arora, D. (2016). Performance comparison of commercial VMM: ESXI, XEN, HYPER-V and KVM. In *Int. Conf. on Computing for Sustainable Global Development* (pp. 1771-1775).

Nadeem, F., & Qaiser, R. (2015). An early evaluation and comparison of three private cloud computing software platforms. Journal of Computer Science and Technology, *30*(3), 639–654. doi:10.1007/s11390-015-1550-1

OpenNebula. (2017, June). OpenNebula: Flexible enterprise cloud made simple. Retrieved from https://opennebula.org/

OpenWall. (2017, June). John the Ripper password cracker. Retrieved from http://www.openwall.com/john/

Pagare, M. J. D., & Koli, D. N. A. (2014). A technical review on comparison of Xen and KVM hypervisors: An analysis of virtualization technologies. *International Journal of Advanced Research in Computer and Communication Engineering*, *3*(12), 2278–1021.

Paradowski, A., Liu, L., & Yuan, B. (2014, June). Benchmarking the performance of openstack and cloudstack. In *2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)* (pp. 405-412). IEEE. doi:10.1109/ISORC.2014.12

Phoronix. (2017, June). Phoronix test suite: Linux testing and benchmarking platform, automated testing, open-source benchmarking. Retrieved from https://www.phoronix-test-suite.com/

Proxmox (2017, Jun). Proxmox: Powerful open source server solutions. Retrieved from https://www.proxmox.com/en/

Rackspace. (2017, June). Rackspace: Managed dedicated and cloud computing services. Retrieved from https://www.rackspace.com/

RedHat. (2017, June). The world's open source leader. Retrieved from https://www.redhat.com/en

Saavedra, R. H., & Smith, A. J. (1996). analysis of benchmark characteristics and benchmark performance prediction. *ACM Transactions on Computer Systems*, *14*(4), 344–384. doi:10.1145/235543.235545

Tran, G. P. C., Chen, Y. A., Kang, D. I., Walters, J. P., & Crago, S. P. (2016, September). Hypervisor performance analysis for real-time workloads. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)* (pp. 1-7). IEEE.

University of Tennessee ICL. (2017, June). CacheBench Home Page. Retrieved from http://icl.cs.utk.edu/llcbench/cachebench.html

Xen Server. (2017, June). XenServer: open source server virtualization. Retrieved from https://xenserver.org/

Younge, A. J., Henschel, R., Brown, J. T., von Laszewski, G., Qiu, J., & Fox, G. C. (2011). Analysis of Virtualization Technologies for High Performance Computing Environments. In *2011 IEEE Int. Conf. Cloud Computing (CLOUD)* (pp. 9–16). doi:10.1109/CLOUD.2011.29

*Mohammad Rafi Ikbal received his master's degree in Computer Science from Alagappa University. He is currently working as high-performance computing administrator at King Abdulaziz University. His research interests are Hypervisors, Parallel programming and computer vision.*

*Roobaea Alroobaea is an assistant professor in College of Computers and Information Technology, Taif University, Kingdom of Saudi Arabia. He is a Chair of support researches and system at Deanship of scientific research in Taif University. He received his bachelor's degree in computer science from King Abdulaziz University (KAU) in Kingdom of Saudi Arabia, in 2008. He achieved a distinction in master's degree from the University of East Anglia in the United Kingdom, in 2011. Additionally, he received his PhD's degree in computer science from the University of East Anglia in the United Kingdom, in 2016. He has published a number of publications, including various journals and conferences. He has been honored by HRH Prince Mohammed bin Nawaf Al Saud, the Saudi ambassador to the UK, in recognition of his research excellence at the University of East Anglia. He is also holder of CCNA certifications*

*Farrukh Nadeem received his Ph.D. Computer Science (with distinction) from the University of Innsbruck, Austria in May 2009. At present, he is serving as an Associate Professor at department of Information Systems, College of Computing and IT, King Abdulaziz University, Jeddah, Saudi Arabia. His main research areas include data science, distributed systems, Cloud computing and decision support systems. He has been involved in a several Austrian and Saudi research and development projects. Dr. Farrukh is author of more than 28 papers, including four book chapters, and 8 journal publications. He holds several distinctions and awards during his educational career. Dr. Farrukh is running multiple Development and Research Projects. He has also developed a Grid Computing infrastructure for HPC at College of Computing and Information Technology.*