Performance Comparison of KVM, VMware and XenServer using a Large Telecommunication Application

Sogand Shirinbab, Lars Lundberg, Dragos Ilie
School of Computing, Blekinge Institute of Technology, Sweden
{Sogand.Shirinbab, Lars.Lundberg, Dragos.Ilie}@bth.se

***Abstract*—One of the most important technologies in cloud computing is virtualization. This paper presents the results from a performance comparison of three well-known virtualization hypervisors: KVM, VMware and XenServer. In this study, we measure performance in terms of CPU utilization, disk utilization and response time of a large industrial real-time application. The application is running inside a virtual machine (VM) controlled by the KVM, VMware and XenServer hypervisors, respectively. Furthermore, we compare the three hypervisors based on downtime and total migration time during live migration. The results show that the Xen hypervisor results in higher CPU utilization and thus also lower maximum performance compared to VMware and KVM. However, VMware causes more write operations to disk than KVM and Xen, and Xen causes less downtime than KVM and VMware during live migration. This means that no single hypervisor has the best performance for all aspects considered here.**

Keywords-Cloud Computing; KVM; Live Migration; VMware vMotion; XenMotion.

I. INTRODUCTION

Virtualization has many advantages over non-virtualized solutions, e.g., flexibility, cost and energy savings [19][34]. As a more specific example, consider the cost associated with test hardware used during professional software development. This includes the initial price for purchasing the equipment, as well as operational costs in the form of maintenance, configuration and consumed electricity. For economic reasons, organizations often choose to use virtualized test servers, so that the test hardware can be shared and maintained in a cost-effective way [20]. In order to provide maximum resource utilization, there should be no restrictions on the mapping of VMs to physical computers, i.e., it should be possible to run a VM on any physical server. In order to balance the load, it is desirable that a VM running on a physical host could be restarted on another physical host, i.e., there is a need for migrating VMs from one physical server to another [21][22][23]. There is support for migration in many commonly used virtualization systems, e.g., KVM Live Migration [16], VMware's vMotion [18] and XenServers's XenMotion [17].

There are three different approaches to VM migration: cold migration, hot migration and live migration. When cold migration is used the guest Operating System (OS) is shut down, the VM is moved to another physical server and then the guest OS is restarted there. Hot migration suspends the guest OS instead of shutting it down. The guest OS is resumed after the VM is moved to the destination host. The benefit of hot migration is that application running inside the guest OS can preserve most of their state after migration (i.e., they are not restarted from scratch). In the live migration approach [13], the VM keeps running while its memory pages are copied to a different host. Live migration reduces the downtime dramatically for applications executing inside the VM. Live migration is thus suitable for high-availability services.

In this paper, we compare the performance of KVM, VMware and XenServer, for two different scenarios: when no VM is migrated, and when a VM is migrated from one physical server to another. The work load is, for both scenarios, a large real-time telecommunication application. In the case when no VM is migrated, we measure the CPU utilization, the disk utilization (the number of write operations), and the average application response time. When a VM is migrated we measure the CPU utilization, the disk utilization (the number of write operations), and the down time due to live migration.

The rest of the paper is organized as follows. In Section II the state of the art is summarized. Section III describes the experimental setup for the different hypervisors, and, in Section IV, we compare and analyze the results for KVM, VMware and XenServer. Finally, related work is discussed in Section V. Section VI concludes the paper.

II. STATE OF THE ART

A. Virtualization

In its simplest form, virtualization is a mechanism for several virtual OS instances on a single physical system. This is typically accomplished using a Hypervisor or Virtual Machine Monitor (VMM), which lies between the hardware and the OS. Virtualization is often beneficial for environments consisting of a large number of servers (e.g., a datacenter).

A virtualization solution relies on several components, such as CPU virtualization, memory virtualization, I/O virtualization, storage virtualization, and so on. In this paper we focus specifically on CPU and memory virtualization.

Current approaches to virtualization can be classified into: full virtualization, paravirtualization and hardware assisted virtualization [11][12].

Full virtualization uses binary translation which translates the kernel code so that privileged instructions can be converted to user-level instructions during run-

 ISBN: 978-1-61208-338-4

time. Detection and translation of privileged instructions typically carries a large performance penalty. KVM and VMware support this approach.

Paravirtualization attempts to alleviate the performance of full virtualization by replacing privileged instructions with specific function calls to the hypervisor, so called hypercalls. This requires changes to the guest OS source code, which is not always possible. In particular, access to the source code of commercial OSs is heavily restricted. Both XenServer and KVM support paravirtualization.

Recent innovations in hardware, particularly in CPU, Memory Management Unit (MMU) and memory components (notably the Intel VT-x and AMD-V architectures [12]), provide some direct platform-level architectural support for OS virtualization. Hardware assisted virtualization offers one key feature: it avoids the need to trap and emulate privileged instructions by enabling guests to run at their native privilege levels. VMware and KVM support this approach.

B. Live Migration

Live migration is a mechanism that allows a VM to be moved from one host to another while the guest OS is running. This type of mobility provides several key benefits, such as fault tolerance, hardware consolidation, load balancing and disaster recovery. Users will generally not notice any interruption in their interaction with the application, especially in the case of non-real-time applications. However, if the downtime becomes too long, users of real-time applications, in particular interactive ones may experience serious service degradation [4].

To achieve live migration, the state of the guest OS on the source host must be replicated on the destination host. This requires migrating processor state, memory content, local storage and network state. The focus of our study is on network state migration.

Pre-copy is the memory migration technique adopted by KVM live migration, vMotion and XenMotion [13][28][27][32]. With this approach, memory pages belonging to the VM are transferred to the destination host while the VM continues to run on the source host. Transferred memory pages that are modified during migration are sent again to the destination to ensure memory consistency. When the memory migration phase is done the VM is suspended on the source host, and then any remaining pages are transferred, and finally the VM is resumed on the destination host [8]. The pre-copy technique captures the complete memory space occupied by the VM (dirty pages), along with the exact state of all the processor registers currently operating on the VM, and then sends the entire content over a TCP connection to a hypervisor on the other host. Processor registers at the destination are then modified to replicate the state at the source, and the newly moved VM can resume its operation [7][27][31].

The Kernel-based Virtual Machine (KVM) is a bare-metal (Type 1) hypervisor. The approach that KVM takes is to turn the Linux kernel into a VMM (or hypervisor). KVM provides a dirty page log facility for live migration, which provides user space with a bitmap of modified pages since the last call [5][6]. KVM uses this feature for memory migration.

VMware is bare-metal (Type 1) hypervisor that is installed directly onto a physical servers without requiring a host OS. In VMware vSphere, vCenter Server provides the tools for vMotion (also known as Live Migration). vMotion allows the administrator to move a running VM from one physical host to another physical host by relocating the contents of the CPU registers and memory [9][10].

XenServer is bare-metal (Type 1) hypervisor and runs directly on server hardware without requiring host OS. XenMotion is a feature supported by XenServer, which allows live migration of VMs. XenMotion works in conjunction with Resource Pools. A Resource Pool is a collection of multiple similar servers connected together in a unified pool of resources. These connected servers share remote storage and common networking connections [1][15][30].

KVM, VMware and XenServer aim to provide high utilization of the hardware resources with minimal impact on the performance of the applications running inside the VM. In this study, we compare their performance by measuring downtime and total migration time during live migration as well as their CPU utilization, when running large telecommunication applications in the VMs.

III. EXPERIMENTAL SETUP

Two HP DL380 G6x86 hosts have been used to test the performance of KVM and VMware ESXi 5.0. On top of the VMware ESXi 5.0, RedHat Enterprise Linux, Version 6.2 has been installed as a guest OS. The same hardware was used to test the performance of Xen for Linux Kernel 3.0.13 running as part of the SUSE Linux Enterprise Server 11 Service Pack 2. Each server is equipped with 24 GB of RAM, two 4-core CPUs with hyperthreading enabled in each core (i.e., a total of 16 logical cores) and four 146 GB disk. Both servers are connected via 1 Gbit Fibre Channel (FC) to twelve 400 GB Serial Attached SCSI (SAS) storage units. All devices are located in a local area network (LAN) as shown in Figure 1.

A. Test Configurations

Three different test setups were evaluated:

- KVM-based setup
- VMware-based setup
- XenServer-based setup

In each setup, two VMs are created inside hypervisor1 and hypervisor2, resulting in a total of four VMs (see Figure 1). One large industrial real-time telecommunication application is installed in the VMs. The application, referred to as server in the reminder of this paper, handles billing related requests. The server instances running on the VMs controlled by hypervisor1 are active in the sense that they are the primary consumers of the requests. The remaining two VMs under the control of hypervisor2 are running one passive instance of the server. Each active server is clustered together with one passive server. Thus, two clusters are created. Both the active and the passive server in a cluster can receive requests. However, all traffic received by the passive server is forwarded to the corresponding active server. The active server then sends the response back to the passive server.

 ISBN: 978-1-61208-338-4

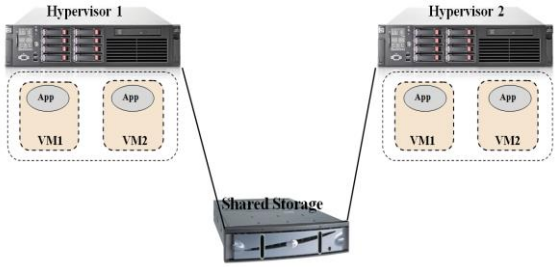

Figure 1. Network Plan

Finally, the passive server sends the response to the requesting system. Traffic going directly to the active server is handled without involving the passive server.
Another separate server runs a simulator that impersonates a requesting system in order to generate load towards the servers running in the clusters. The simulator is also located in the same LAN, but is not shown in Figure 1.

B. Test Cases

Two kinds of tests are considered in this study: performance tests and live migration tests.

1) Performance tests

In these tests, we vary the number of CPU cores (logical cores) in the VMs as well as the load towards the application.

We have three different core configurations: 6, 12 and 16 cores. For test cases with 12 cores and 16 cores the RAM for the VM is set to 24 GB, but for test case with 6 cores, the RAM size set to 14 GB for each of the VMs. This is an application specific setting that is recommended by the manufacturer. A single cluster is used for the case with 12 and 16 cores, respectively. Both clusters are used when testing the 6 cores configuration in order to assess the performance of two 6-core systems versus the performance a single 12-core system.

There are five load levels used in this test: 500, 1500, 3000, 4300, and 5300 incoming requests per second (req/s).

For each setup the following metrics are measured: CPU utilization, disk utilization and response time.

CPU utilization and disk utilization are measured inside the hypervisor on both servers using the commands presented in Table I. For disk utilization, we consider only write operations to the shared storage shown in Figure 1. The response time is measured inside the simulator as the duration from the instant a request is sent from the simulator to the application until the simulator receives the corresponding reply.

2) Live Migration tests

In these tests, we measure CPU and disk utilization during live migration. Four VMs with 6 cores CPU and 14 GB of RAM were created. For each configuration, a single VM (active server, e.g., VM1 on Hypervisor1 in Figure 1) is migrated from the source host to the destination host while the simulator creates a load of 100 req/s for the VM. At the same time the other VM (e.g. VM2 on Hypervisor1 in Figure 1) on the source host is receiving 1500 req/s. The other VMs (VM1 and VM2 on Hypervisor2 in Figure 1) on the destination host receive negligible traffic in the form of 100 req/s and thus are not completely idle.

TABLE I. CPU and DISK UTILIZATION COMMAND API

Virtualization System	Command Interface	
	CPU Utilization	***Disk Utilization***
KVM	ssh + sar	ssh + iostat
VMware	vCenter Server-performance graphs	vCenter Server-performance graphs
XenServer	ssh + xentop	ssh + iostat
Non-virtualized Server	ssh + sar	ssh + iostat

The application manufacturer considered this as a realistic example when one would like to migrate a VM to load-balance the system.

In addition to CPU and disk utilization, we measure the downtime and the total migration time. The total migration time is obtained from the hypervisor for KVM and XenServer, and from vCenter for VMware (see Table I). Downtime is defined as the time from the instant when the VM is suspended on the source host (Hypervisor1 in Figure 1) until the VM is restarted on the destination host (Hypervisor2 in Figure 1). We measured the downtime inside the simulator and our results indicate that it corresponds to the maximum response time of the application.

IV. COMPARISON BETWEEN KVM, VMWARE AND XENSERVER

In Section IV-A, the KVM, VMware and XenServer virtualization systems are compared in terms of CPU utilization (6 cores, 12 cores, and 16 cores), disk utilization and response time. These values have been measured for different loads (500, 1500, 3000, 4300, 5300 req/s) except for XenServer, which could not handle the highest load (5300 req/s).In Section IV-B, we compare the CPU utilization and the disk utilization during live migration, and in Section IV-C, we compare the total migration time and downtime of the VMs during live migration for the KVM, VMware and Xen Server setups, respectively.

A. CPU, Disk Utilization and Response Time (6 cores, 12 cores, 16 cores)

CPU and disk utilization are measured inside the hypervisors. We also performed the same measurements on the non-virtualized (target) server in order to establish a baseline for our results (see Table I). The response time is measured in the simulator.

As shown in Figure 2, Xen has the highest CPU utilization (approximately 80%) in the test case with 16 cores. Because of this high CPU utilization the application failed for traffic loads higher than 4300 req/s. KVM and VMware CPU utilization increases proportional to the load with an increase rate similar to that of the target. In Figure 3, we can observe that again Xen CPU utilization is significantly higher compared to VMware, KVM and the target in case of 12 cores. As shown in Figure 4, KVM, VMware and the target CPU utilization in case of 6 cores, are almost identical while Xen CPU utilization is the highest and at the highest point is around 70% which is the 20% higher CPU utilization compared to KVM, VMware and the target.

In Figure 5, we can observe that in case of 16 cores, VMware has the highest disk utilization, up to 25000 KB/s. KVM and Xen the disk utilization is linearly

 ISBN: 978-1-61208-338-4

increasing with a rate similar to that of the disk utilization of the target. However, for KVM's and Xen's disk utilization is always around 2000 KB/s higher compared to the target. As shown in Figure 6, in case of 12 cores, Xen and KVM disk utilization is 5000 KB/s higher compared to the target while disk utilization for VMware is the highest, with a maximum around 30000 KB/s. In Figure 7, we can observe that VMware has the highest disk utilization compared to KVM and Xen, which show 34000 KB/s at the highest point. Xen's disk utilization in case of 6 cores is higher than KVM. The maximum disk utilization for Xen is around 25000 KB/s while the maximum KVM disk utilization is around 20000 KB/s. That is 5000 KB/s higher compared to the target but still the lowest compared to other virtualization systems.

Figure 8 shows that the response time of the application when using Xen is the highest for all traffic loads except for loads higher than 4300 req/s. Since for loads higher than 4300 req/s the application failed when using Xen, KVM has the highest response time after Xen, and at the highest point is around 25 ms in case of 16 cores. The response times of the application when using VMware is similar to the response times we had on the target. As shown in Figure 9, the response time of the application when using Xen reaches 26 ms at the highest point. In case of KVM the application has also high response times with a maximum of around 20 ms, which is higher than VMware's. The application response times when using VMware is similar to the response times of the application on the target. In Figure 10, we can observe that response time of the application when using Xen at the highest point is more than 25 ms, which is twice the application response time in case of the non-virtualized target. In the case of the 6 cores configuration using KVM the response time increases with a similar rate to the case when using VMware. However, for KVM and VMware the response times are around 5 ms higher compared to the target.

B. CPU, Disk Utilization and ResponseTime during Live Migration

CPU utilization is measured inside the hypervisors on both the source and the destination servers, during the live migration. Disk utilization is also measured inside both hypervisors. We initiate a migration after the system has been running for 15 minutes.

As shown in Figure 12, KVM's CPU utilization on the source is around 26% before the live migration begins. The CPU utilization on the destination is around 6%. After the live migration has been started, the CPU utilization first increases to 35% and then decreases to 18% on the source. However, on the destination server the CPU utilization settles around 13% after the live migration. As shown in Figure 12, VMware's CPU utilization before live migration is around 20% on the source hypervisor and around 4% on the destination hypervisor. When the live migration has been started, the CPU utilization on source increases to about 34% and remains at that level during the live migration. On the destination, the CPU utilization becomes around 15% after the live migration has started. After the live migration has stopped, the CPU utilization decreases to around 15% on the source hypervisor and to around 10% on the destination hypervisor. In Figure 12, we can observe that the Xen CPU utilization before live migration is around 34% on the source hypervisor and around 7% on the destination hypervisor. In the beginning of the live migration, the CPU utilization on source increases to around 40% and on the destination the Xen CPU utilization increases to around 13%. After the live migration is completed, the CPU utilization on the source decreases to around 29%, while on the destination's CPU utilization increases to 15%.

As shown in Figure 13, KVM's disk utilization on the source is around 10000 KB/s before live migration. On the destination, the disk utilization is around 6000 KB/s before live migration. After the live migration has started, the disk utilization on the source decreases to 9000 KB/s, while on the destination's disk utilization increases to 7000 KB/s. As shown in Figure 13, VMware's disk utilization is around 15000 KB/s on the source before live migration while on the destination the disk utilization is around 7000 KB/s. After the live migration, the disk utilization on the source decreases to around 13000 KB/s and on the destination it increases to around 9000 KB/s. In Figure 13 we can observe that the Xen disk utilization before the live migration is around 13000 KB/s on the source and around 6000 KB/s on the destination. When the live migration has started, the disk utilization increases to around 30000 KB/s on the source and to around 23000 KB/s on the destination. After the live migration has completed, the disk utilization on the source decreases to around 9000 KB/s and on the destination the disk utilization increases to around 10000 KB/s.

C. DowntimeandTotal Migration Time

The downtime has been obtained from the maximum response time, which is measured inside simulator during the live migration. Downtime corresponds to the time that application is not available and the VM is suspended.

As shown in Figure 11, the response time of the application when using KVM as hypervisor is around 1 ms before the live migration is started, but when the VM is suspended the response time increases to 700 ms. So the application was down for less than 700 ms. In Figure 11, we can observe that the response time of the application when using VMware as hypervisor is around 1 ms, but when the VM is totally down the application response time increases to 3000 ms. So the application downtime was around 3000 ms. As shown in Figure 11, before the live migration starts the application response time when using Xen is around 4 ms. When the live migration begins, the response time increases to 280 ms. So the application was down for less than 4 ms.

The total migration time is calculated inside the source hypervisor. It corresponds to the time that the VM started to be migrated until the complete VM state has been transferred to the destination hypervisor (see Figures 12-13). The total migration time for VMware, KVM and Xen is around 2 minutes.

V. RELATED WORK

In recent years, there have been several efforts to compare different live migration technologies. Xiujie et al. [1] compare the performance of vMotion and XenMotion under certain network conditions defined by varying the available bandwidth, link latency and packet

loss. Their results show that vMotion produces less data traffic than XenMotion when migrating identical VMs. However, in networks with moderate packet loss and delay, which are typical in a Virtual Private Network (VPN), XenMotion outperforms vMotion in total migration time.

Tafa et al. [2] compare the performance of three hypervisors: XEN-PV, XEN-HVM and Open-VZ. They simulated the migration of a VM using a warning failure approach. The authors used a CentOS tool called "Heartbeat" that monitors the well-being of high-availability hosts through periodic exchanges of network messages. When a host fails to reply to messages the tool issues a failure notification that causes the hypervisor to migrate the VM from the "dead" host to one that is "alive". Further, they compared CPU usage, memory utilization, total migration time and downtime. The authors have also tested the hypervisor's performance by changing the packet size from 1500 bytes to 64 bytes. From these tests they concluded that Open-VZ has a higher CPU usage than XEN-PV, but the total migration time is smaller for Open-VZ (3.72 seconds for packet size of 64 bytes) than for XEN-PV (5.12 seconds for packet size of 64 bytes). XEN-HVM has lower performance than XEN-PV; especially regarding downtime. XEN-HVM had16 ms downtime while XEN-PV had 9 ms downtime for packet size of 64 bytes compared to our results with the large application we have got 300 ms downtime for Xen and total migration time of around 2 minutes.

In Chierici et al. [3] and Che et al. [29] present a quantitative and synthetically performance comparison between Xen, KVM and OpenVZ. They used several benchmarks (NetIO, IOzone, HEP-Spec06, Iperf and bonnie++) to measure CPU, network and disk accesses. According to their measurements, the OpenVZ has the best performance; also Xen hypervisor offers good performance while KVM has apparently low performance than OpenVZ and Xen.

There has been a similar study to our work carried out by Hicks, et al. [14], in which the authors focused only on memory migration and storage migration in the KVM, XenServer, VMware and Hyper-V virtualization systems. However, they did not consider CPU utilization of hypervisor during live migration in their study.

Clark et al. [27] introduced a method for the migration of entire operating system when using Xen as a hypervisor. They have tested different applications and recorded the service downtime and total migration time. Their results show 210 ms downtime for SPECweb99 (web-server) and 60 ms downtime for Quake3 (game server) during the migration.

Du et al. [24] proposed new method called Microwiper which makes less dirty pages for live migration. They implemented their method on the pre-copy based live migration in Xen hypervisor. They've tested two different programs with one with fixed memory writes and the other one with very quick memory writes. They compared the downtime and total migration time when using their method (Microwiper) versus the original Xen live migration (XLM). Their results show the original Xen live migration gets 40 ms downtime for VM memory size of 1024 MB when running quick memory writes program and total migration time of 11 seconds while their technique (Microwiper) decreases the downtime so it became around 10 ms but they got the same total migration time.

Web 2.0 application [33] has been evaluated by Voorsluys et al. [25] in terms of downtime and total migration time during live migration. They run XenServer as a hypervisor on their VM hosts. According to their experiments downtime of their system when serving 600 concurrent users is around 3 seconds and their total migration time is around 44 seconds which is much higher compared to our results because of the application that they've used also their setup is different.

Jo et al. [26] implemented a technique to reduce the duplication of data on the attached storage. They used different applications, RDesk I and II, Admin I, etc. and they measured the down time and total migration time during live migration when using XenServer as hypervisor. Their experiment shows 350 seconds total migration time for the original Xen live migration when the maximum network bandwidth is 500 megabits per second while using their proposed technique reduces this number to 50 seconds when duplication ratio is up to 85 percent.

VI. CONCLUSION AND FUTURE WORK

The results of the performance tests for different configurations of number of CPU cores show that KVM and VMware CPU utilization is almost identical and similar to CPU utilization on the target machine (non-virtualized) while XenServer has the highest CPU utilization with a maximum around 80%. In terms of disk utilization, the results indicate that KVM and Xen have similar disk utilization while VMware has the highest disk utilization (around 30000 KB/s for the highest load). The response time of the application is the highest when using Xen as hypervisor showing around 25 ms at the highest point. For KVM and VMware, the response time is almost similar (around 20 ms).

In general, KVM and VMware perform better in terms of CPU utilization while Xen CPU utilization is the highest. In terms of disk utilization KVM and Xen have similar performance while VMware has the highest disk utilization. Further, in terms of response time Xen has the longest response times compared to KVM and VMware.

As the results have shown, the CPU utilization during live migration is lower for KVM than for VMware while Xen had the highest CPU utilization during live migration. The disk utilization when KVM is used is 1000 KB/s lower compared to VMware during the migration.

For VMware, the downtime is measured to 3 seconds during live migration. For KVM and Xen the measured downtime are only 0.7 seconds and 0.3 seconds, respectively.

In general, the results presented in this study show that both VMware and KVM perform better in terms of application response time and CPU utilization for a configuration of two VMs with 6 cores each, compared to a configuration with a single VM with 16 or 12 cores. Xen's performance is below that of the two other virtualization systems tested. However, Xen's live migration technology, XenMotion, performs better than VMware's vMotion and KVM live migration technology in terms of downtime.

 ISBN: 978-1-61208-338-4

REFERENCES

[1] F. Xiujie, T. Jianxiong, L. Xuan, and J. Yaohui, "A Performance Study of Live VM Migration Technologies: vMotion vs XenMotion," Proceedings of the International Society for Optical Engineering, Shanghai, China, 2011, pp. 1-6.

[2] I. Tafa, E. Kajo, A. Bejleri, O. Shurdi, and A. Xhuvani, "The Performance between XEN-HVM, XEN-PV And OPEN-VZ During Live Migration," International Journal of Advanced Computer Science and Applications, 2011, pp. 126-132.

[3] A. Chierici and R. Veraldi, "A Quantitative Comparison between Xen and KVM," 17th International Conference on Computing in High Energy and Nuclear Physics, Boston, 2010, pp. 1-10.

[4] D. Huang, D. Ye, Q. He, J. Chen, and K. Ye, "Virt-LM: a benchmark for live migration of virtual machine," ACM SIGSOFT Software Engineering Notes, USA, 2011, pp. 307-316.

[5] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: the linux virtual machine monitor," OLS, Ottawa, 2007, pp. 225-230.

[6] Red Hat. (2009). "KVM- Kernel based virtual machine,"[Online]. Available from: http://www.redhat.com/rhecm/rest-rhecm/jcr/repository/collaboration/jcr:system/jcr:versionStorage/5e7884ed7f00000102c317385572f1b1/1/jcr:frozenNode/rh:pdfFile.pdf
2014-03-10

[7] M.R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," international conference on virtual execution environments, USA, 2009, pp. 51-60.

[8] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmorth, "Evaluation of delta compression techniques for efficient live migration of large virtual machines," 7th ACM SIGPLAN/SIGOPS International conference on Virtual execution environments, USA, 2011, pp. 111-120.

[9] S. Lowe, "Mastering VMware vSphere 5," Book, 2011.

[10] E. L. Haletky, "VMware ESX and ESXi in the Enterprise:Planning Deployment of Virtualization Servers," Upper Saddle River, NJ: Prentice Hall, 2011.

[11] A.Kovari and P.Dukan, "KVM & OpenVZ virtualization based IaaS Open Source Cloud Virtualization Platform," 10th Jubilee International Symposium on Intelligent Systems and Informatics, Serbia, 2012, pp. 335-339.

[12] A.J. Younge, R. Henschel, and J.T. Brown, "Analysis of Virtualization Technologies for High Performance Computing Environments," 4th IEEE International Conference on Cloud Computing, Washington, DC, 2011, pp. 9-16.

[13] A. Warfield, et al., "Live Migration of Virtual Machines," Proceedings of the 2nd conference on Symposium on Network Systems Design and Implementation, USA, 2005, pp. 273-286.

[14] A. Hicks, et al., "A Quantitative Study of Virtual Machine Live Migration," Proceedings of the ACM Cloud and Autonomic Computing Conference, USA, 2013, pp. 1-10.

[15] J. Wang, L. Yang, M. Yu, and S. Wang, "Application of Server Virtualization Technology Based on Citrix XenServer in the Infromation Center of the Public Security Bureau and Fire Service Department," Proceedings of the Computer Science and Society, Kota Kinabalu, 2011, pp. 200-202.

[16] KVM. [Online]. Available from: http://www.linux-kvm.org/page/Main_Page
2014-03-10

[17] XenServer. [Online]. Available from: http://www.citrix.com/products/xenserver/overview.html
2014-03-10

[18] VMware. [Online]. Available from: http://www.vmware.com/
2014-03-10

[19] Ch. Cai and L. Yuan, "Research on Energy-Saving-Based Cloud Computing Scheduling Strategy," Journal of Networks, 2013, pp. 1153-1159.

[20] E.Michael and F. Janos, "A Survey of Desktop Virtualiztion in Higher Education: An Energy-and Cost-Savings Perspective," 19th Americas conference on Information Systems, 2013, pp. 3139-3147.

[21] X. Li, Q. He, J. Chen, K. Ye, and T. Yin, "Informed Live Migration Strategies of Virtual Machines for Cluster Load Balancing" Proceedings of the 8th IFIP International Conference, 2011, pp. 111-122.

[22] Z. Wenyu, Y. Shaoubao, F.Jun, N. Xianlong, and S. Hu, "VMCTune: A Load Balancing Scheme for Virtual Machine Cluster Based on Dynamic Resource Allocation" Proceedings of the 9th International Conference on Grid and Cloud Computing, 2010, pp. 81-86.

[23] P. Riteau, C. Morin, and T. Priol, "Shrinker: Efficient Live Migration of Virtual Machines" Concurrency and Computation: Practice and Experience, 2013, pp. 541-555.

[24] Y. Du, H. Yu, G. Shi, J. Chen, and W. Zheng, "Microwiper: Efficient Memory Propagation in Live Migration of Virtual Machines," 39th International Conference on Parallel Computing, 2010, pp. 141-149.

[25] W. Voorsluys, J. Broberg, S.Venugopal, and R. Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," Proccedings of the 1st International Conference on Cloud Computing, 2009, pp. 254-265.

[26] Ch. Jo, E. Gustafsson, J. Son, and B. Egger, "Efficient Live Migration of Virtual Machines Using Shared Storage," Proceedings of the 9th International Conference on Virtual Execution Environments, 2013, pp. 41-50.

[27] C. Clark, et al., "Live Migration of Virtual Machines," Proceedings of the 2nd symposium on Networked Systems Design and Implementation, 2005, pp. 273-86.

[28] S. Akoush, R. Sohan, A. Rice, A.W. Moore, and A. Hopper, "Predicting the Performance of Virtual Machine Migration," Proceedings of the 18th IEEE/ACM international symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, 2010, pp. 37-46.

[29] J. Che, Y. Yu, C. Shi, and W. Lin, "A Synthetical Performance Evaluation of OpenVZ, Xen and KVM," Proceedings of the IEEE conference on Asia-Pacific Services Computing, 2010, pp. 587-594.

[30] S. Kikuchi and Y. Matsumoto, "Impact of Live Migration on Multi-tier Application Performance in Clouds," Proceedings of the 5th IEEE international conerence on Cloud Computing, 2012, pp. 261- 268.

[31] H. Liu, H. Jin, Ch. Xu, and X. Liao, "Performance and Energy Modelling for Live Migration of Virtual Machines," Proceedings of the conference on Cloud Computing, 2013, pp. 249-264.

[32] S. Kikuchi and Y. Matsumoto, "Performance Modelling of Concurrent Live Migration Operations in Cloud Computing Systems using PRISM Problemabilistic Model Checker," Proceedings of the IEEE 4th international conference on Cloud Computing, 2011, pp. 49-56.

[33] L. Wang, et al., "Cloud Computing: a Prespective Study," Proceedings of the New Generation Computing conference, 2010, pp. 137-146.

[34] J. Che, Q. He, Q. Gao, and D. Huang, "Performance Measuring and Comparing of Virtual Machine Monitors," Proceedings of the 5th interantional conference on Embedded and Ubiquitous Computing, 2008, pp. 381-386.

 ISBN: 978-1-61208-338-4

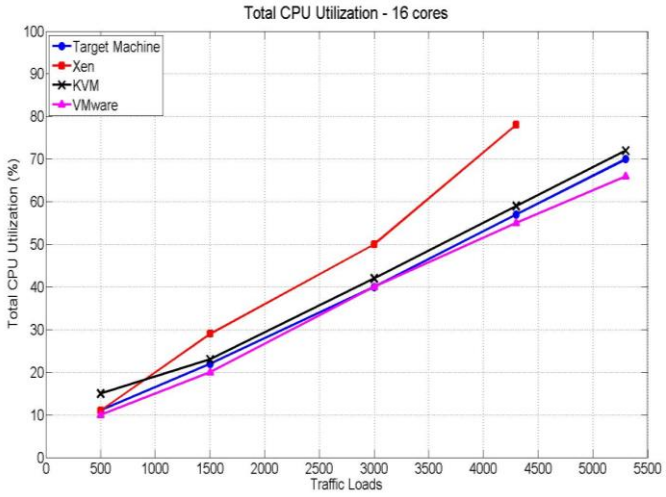

Figure 2. KVM, VMware and Xen CPU utilization for 16 cores

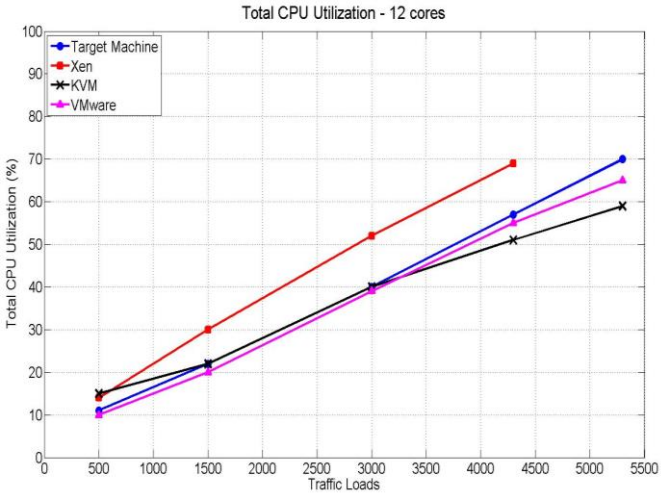

Figure 3. KVM, VMware and Xen CPU utilization for 12 cores

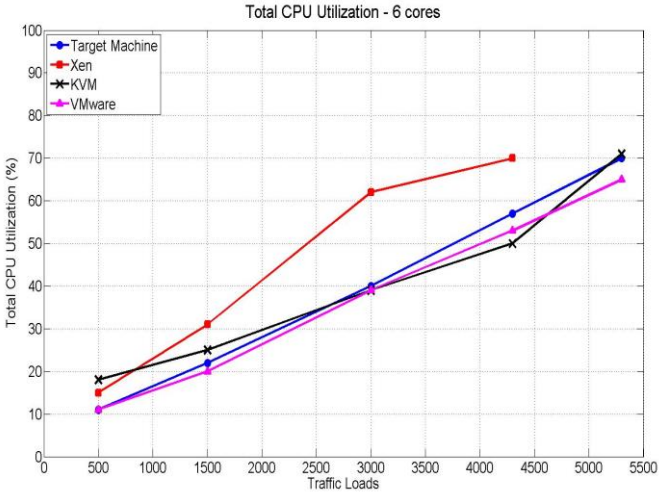

Figure 4. KVM, VMware and Xen CPU utilization for 6 cores

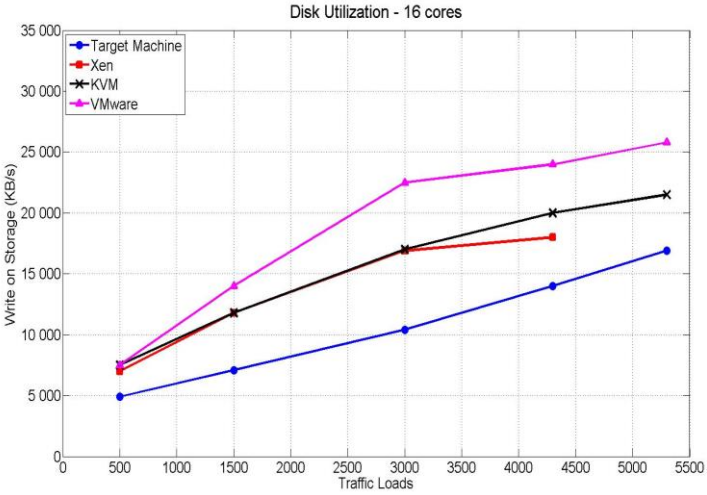

Figure 5. KVM, VMware and Xen disk utilization for 16 cores

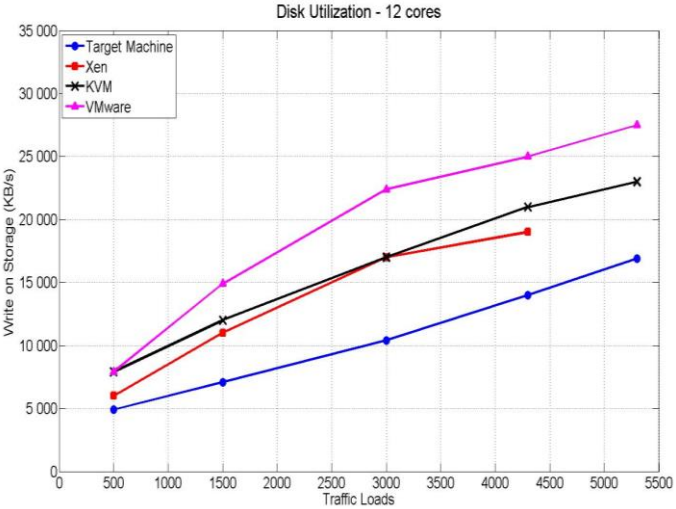

Figure 6. KVM, VMware and Xen disk utilization for 12 cores

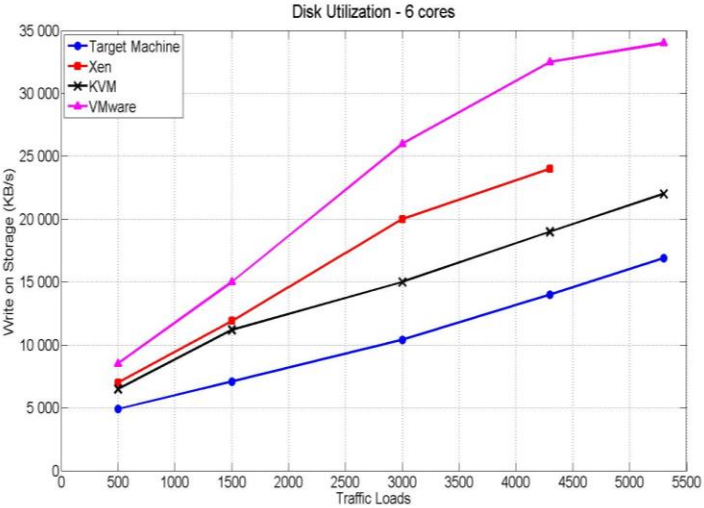

Figure 7. KVM, VMware and Xen disk utilization for 6 cores

 ISBN: 978-1-61208-338-4

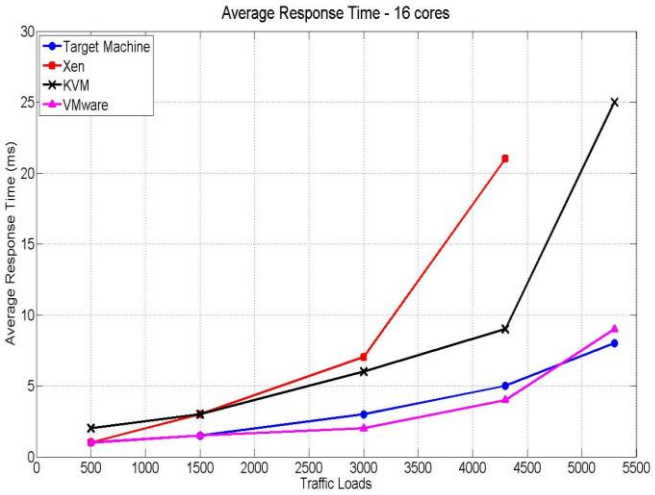

Figure 8. KVM, VMware and Xen response time for 16 cores

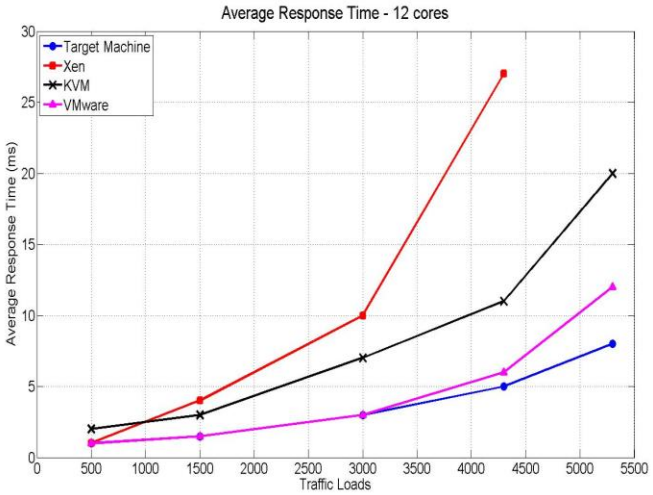

Figure 9. KVM, VMware and Xen response time for 12 cores

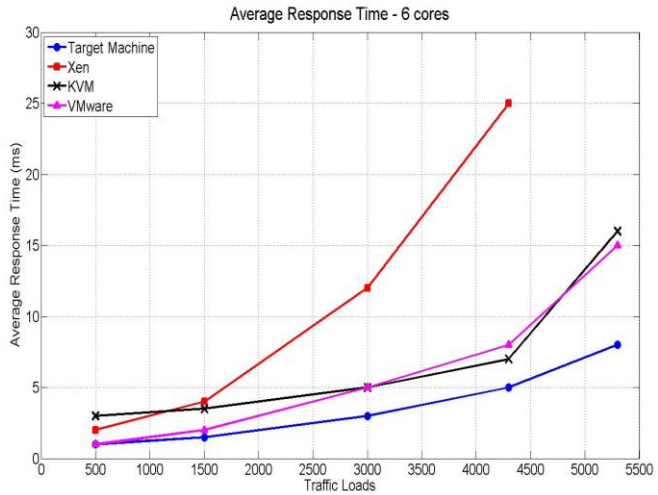

Figure 10. KVM, VMware and Xen response time for 6 cores

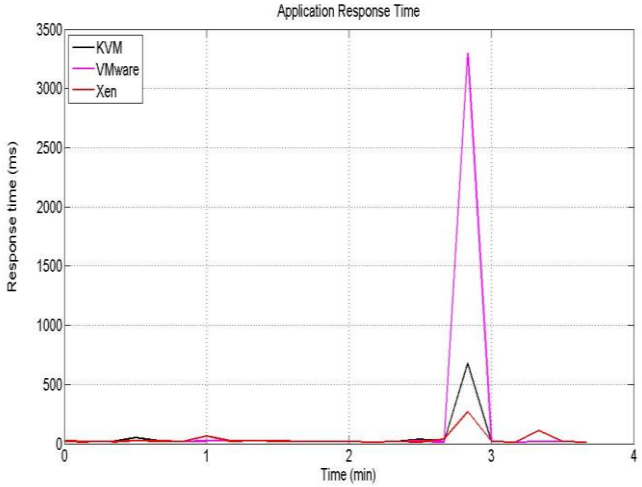

Figure 11. KVM , VMware and Xen response time during live migration

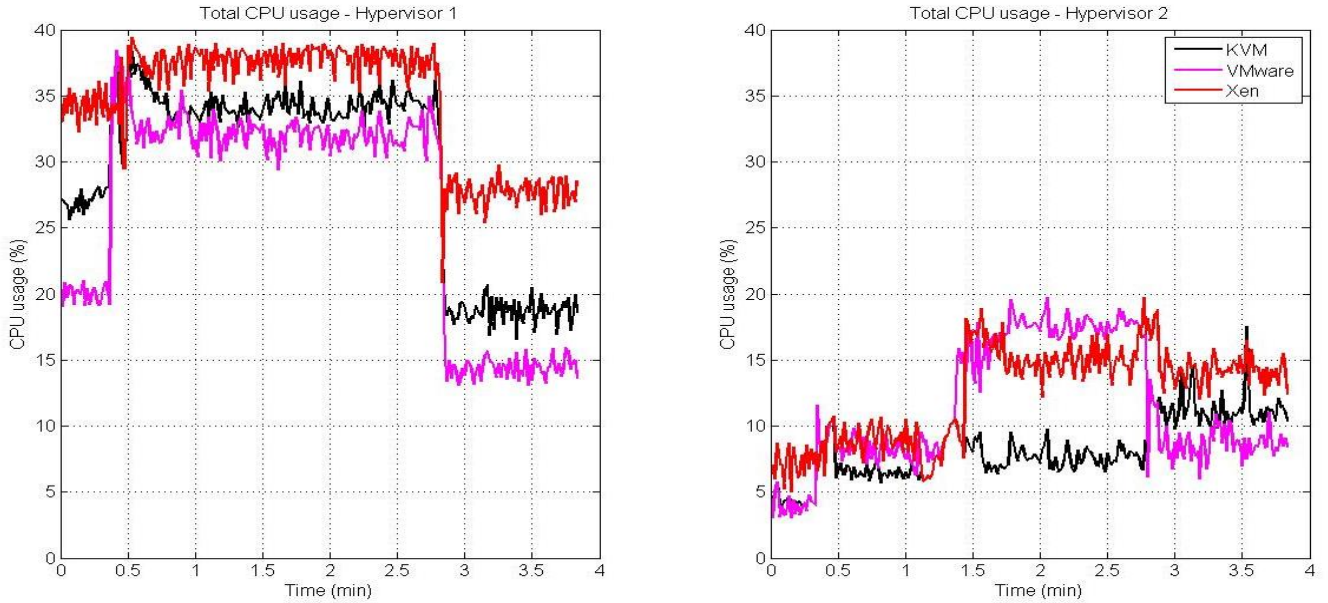

Figure 12. KVM , VMware and Xen CPU utilization during live migration

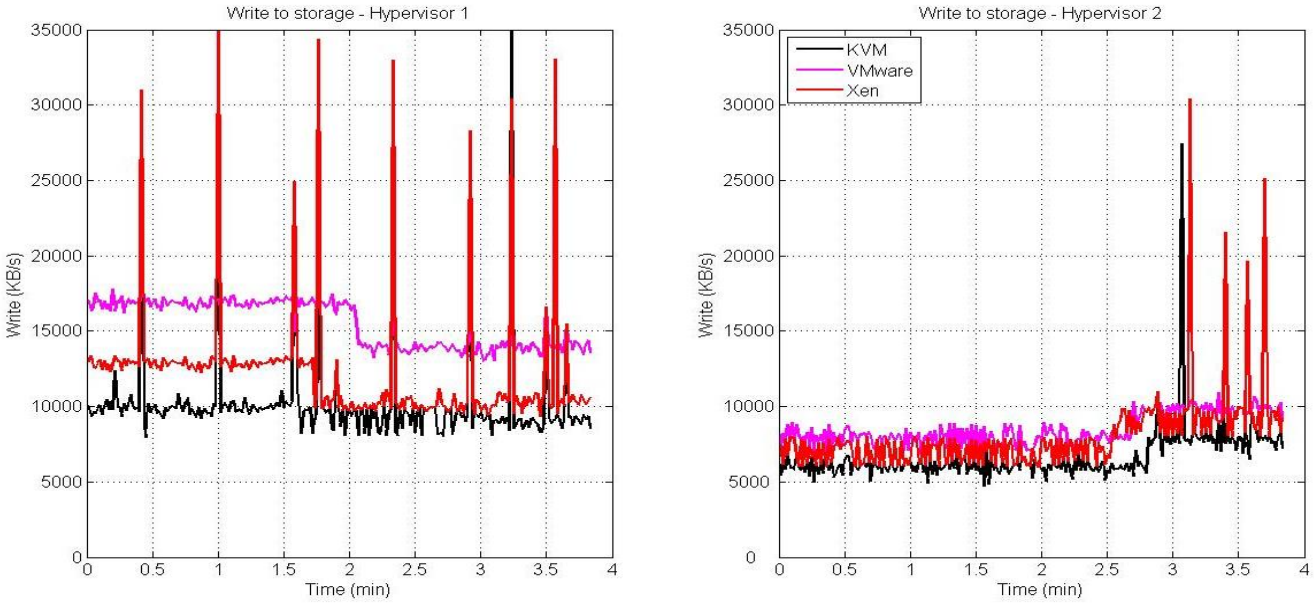

Figure 13. KVM , VMware and Xen disk utilization during live migration