Performance comparison of native host vs. ESXi hypervisor-based virtualization

Borislav Đorđević, *Member, IEEE*, Srđan Milenković, Nikola Davidović and Valentina Timčenko, *Member, IEEE*

Abstract – The main objective of this paper is performance comparison of hypervisor-based virtualization with VMware ESXi virtual machines and native host machine. From all performance classes, for the needs of this research we have chosen the evaluation of the file system performance. The measurements are carried out under equivalent conditions and by a unique test method, using the Filebench software, which guarantees equality and independence from the impact of hardware and operating system characteristics. As the base operating system we have used CentOS 7.7 with the latest updates, while ESXi 6.7 was used as the hypervisor. Performances are compared for the native host machine and ESXi server with one, two and three virtual machines (VM) running simultaneously. We have also analysed the expected behaviours, verified the assumption with Filebench testing software, and provided the concluding remarks for this papers research topic.

Key words – Virtualization; Filebench; Hypervisors; ESXi; VMware; CentOS; Virtual Machines

I. Introduction

In IT world, the term virtualization refers to the act of creating a virtual version of something, or it is the process of creating and running a virtual instance of a computer resource in a layer abstracted from the actual hardware. It is used to describe virtual computer hardware platforms, storage devices, network resources, server infrastructure, etc. We can experience virtualization in almost all segments of today's computer technology. The main idea behind virtualization is a very simple and came from the corporative approach: the need to satisfy the increase in the utilization of available hardware resources, while at the same time reducing the costs of the infrastructure. Virtualization did exist as a technology even some 30 years ago, but the hardware of those days could not exploit the full usage that virtualization brought, so it was disregarded until progress was made in computer technology giving to virtualization a new meaning, shaping it to what it looks today. Nowadays, thanks to this technology it is possible to run multiple independent operating systems on one physical server. Some of the benefits that virtualization provides are primarily related to saving the necessary physical space that would be needed for the accommodation of the devices and also the electrical energy consumption that would inevitably be used for powering such devices. Today, the use of virtualization in a simple way increases server availability and isolation, making it one of main reasons why these technologies are so popular [1]. When using these technologies it is important to mention that the level of hardware utilization of servers without virtualization is in the range of 15% of its maximum capacity, while with the use of virtualization technologies the utilization raises to more than 70%. These technologies however come with a price, or to be exact, with the retention or even increasing availability of resources, while it is realistic to expect a somewhat lower performance of virtualized systems when compared to the non-virtualized systems, which is the main topic of this paper.

There are several virtualization types: virtualization of hardware, software, desktop, data, network, memory, storage, etc. We are focused on hardware virtualization. Hardware virtualization implies the use of a hypervisor, a layer that acts as a mediator between the host and virtual machine, which is nothing more than a simulated computing environment that can, but does not have to be equal to the physical environment that it simulates. In addition to the classification by the location of the hypervisor layer, the hardware virtualization also depends on what type of virtualization is provided, and can be categorized as: full, hardware-assisted, and paravirtualization.

Full (native) virtualization is a virtualization technique that completely simulates the underlying hardware. Hardware-assisted virtualization (Intel VT-x or AMD-V) is platform virtualization approach that enables efficient full virtualization using help from hardware capabilities, primarily from host processors. In this situation, the processor simulates hardware that does not have to be the same as physical. Paravirtualization is an enhancement of virtualization technology in which a guest operating system is modified prior to the installation inside a virtual machine in order to allow all guest OS within the system to share resources and successfully collaborate, rather than attempt to emulate an entire hardware environment [2].

The remainder of this paper will be structured as follows. Section II provides a brief description of the technologies that are mentioned in the paper and a short review of related work for this project. Section III provides the description of the

Borislav Đorđević – Institute Mihailo Pupin, Volgina 15, 11000 Belgrade, Serbia, (borislav.djordjevic@pupin.rs)

Srđan Milenković - School of Electrical and Computer Engineering of Applied Studies, Vojvode Stepe 283, 11000 Belgrade, Serbia, (smilenkovic1992@gmail.com)

Nikola Davidović – University of East Sarajevo, Faculty of Electrical Engineering, Vuka Karadzica 30, 71123 East Sarajevo, RS, Bosnia and Herzegovina, (nikola.davidovic@etf.ues.rs.ba)

Valentina Timčenko - Institute Mihailo Pupin, School of Electrical Engineering, Belgrade, Serbia, (valentina.timcenko@pupin.rs)

performance-measuring tool that we have used for this experiment. In Section IV, we present a short description of the architecture of the used hypervisor. Section V presents the hypothesis and methodology used to achieve performance comparison. In Section VI, we present the test environment and configuration for the experiment. Test results for our benchmarks' tests are presented in Section VII. In Section VIII, we draw conclusions to the work made in this paper.

II. Related work and objective

This paper is primarily devoted to analysis of the performances of hypervisor-based virtualization with one of the most commonly used hypervisors. The hypervisor serves as a layer between the virtual machine's operating system and the host's physical memory, providing data integrity and isolation of VMs. Thanks to hardware-assisted virtualization which is accomplished via EPTS (extended page tables, for Intel chipsets) or RVI (rapid virtualization indexing, for AMD) we have a large increase of speed compared to software memory virtualization [3]. The paper considers advantages of using virtual machines while creating modern network infrastructure; as well as describes an experiment using common test environments and programs for measuring and analysis of hypervisors and their performances. Benchmarking is a popular approach nowadays for many devices and general I/O performance analysis, whereas the special attention is put on the problem of fast input/output support [4-6].

Main contribution of this paper is the examination of the performances of the native host operating system and hypervisor-based virtualization of VMware ESXi [7] [8]. As the technology that was used for this research is relatively new, there are not many references in literature that research with similar environments, tools, and test characteristics. The goal of this paper is to examine the file system performance of the generated workload through Filebench software tool for: (1) mail server scenario which is dominated by random read and random write components; (2) web server scenario where random read components dominate; (3) file server scenario in which both random and sequential components are equally represented; and (4) random file access scenario dominated by random read component [9].We have set up a model for the file system performance analysis of the native host and ESXi based virtual machines. The results of this experiment should give us a full picture of how the performance of a native machine compares to the performance of a hypervisor-based virtual machine.

III. Filebench

Filebench is a software test environment (usually called a benchmark) used to measure the performance of various parts of an operating system. What sets Filebench apart from other benchmarks is the fact that it is equipped with several predefined workloads, which allows users to easily test their systems in various forms (most popular forms being a mail server or a file server) [10]. Presently many benchmarks hard code the workloads they generate quite rigidly, meaning that a user can specify some of the basic workload parameters, but cannot really control the execution flow of the workload in detail. Filebench gives its users freedom to define workloads using a Workload Model Language (WML). WML is mainly composed of four main parts: fileset, process, thread, and flowop.

A standard Filebench test is executed in two stages: fileset pre-allocation and a workload execution. First part of any workload execution is defining a fileset that it uses. A fileset is a named collection of files and to define it a user must specify its name, path, number of files, and a few other optional attributes that can be included in a filesets creation. After defining a fileset the next step are the processes in WML that represent real UNIX processes which are created by Filebench during the test. Every process is made of one or more threads representing an actual POSIX threads and every thread executes a loop of flowops. A single flowop is a representation of a file system operation that is translated to a system call by Filebench.

The ending of a WML file usually contains one of two "run" commands (run and psrun) that tell Filebench to allocate the defined filesets, prepare the required number of UNIX processes and threads, and start a cycled flowops execution. After completing a run, Filebench gives a number of different metrics, where the most important one for the user is operations per second. This is the total number of executed flowop instances (in all processes and threads) divided by the time it took for a full run of the workload. To generate a workload and start the measurement of a particular part of the system, one must execute the *filebench -f workload.f* command.

IV. ESXi Hypervisor

VMware ESXi (Elastic Sky X "integrated") is a type-1 hypervisor developed by VMware for deploying and serving virtual machines that was made from its predecessor ESX. Type-1 hypervisors run directly on the host's hardware to control the given hardware and to manage guest operating systems, and for this reason they are mainly called bare metal hypervisors (Figure 1). A guest operating system runs on another level above the hypervisor.

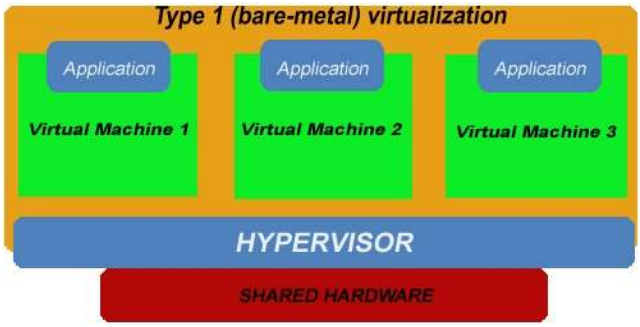

Fig. 1.Type-1 (bare metal) hypervisor

VMware ESXi is a hypervisor that runs on the host server hardware without the underlying operating system. ESXi provides a virtualization layer that abstracts the CPU, storage,

memory and networking resources of the physical host into multiple virtual machines. That means that applications running in virtual machines can access these resources without direct access to the underlying hardware. VMware refers to the hypervisor used by VMware ESXi as VMkernel and it receives requests from virtual machines (as processes that run on top of it) for resources and presents the requests to the physical hardware [12-14]. The kernel also provides means for running all processes on the system, including management applications and agents as well as virtual machines. It has control of all hardware devices on the server, and manages resources for the applications as shown in Figure 2 [15]. The main processes that run on top of VMkernel are:

- Direct Console User Interface (DCUI) — the low-level configuration and management interface, accessible through the console of the server, used primarily for initial basic configuration.
- The VMM, virtual machine monitor, which is the process that provides the execution environment for a virtual machine, as well as a helper process known as VMX. Each running virtual machine has its own VMM and VMX process.
- Various agents are used to run and enable high-level VMware Infrastructure management from remote applications.
- The Common Information Model (CIM) system is the interface that enables hardware-level management from remote applications via a set of standard APIs.

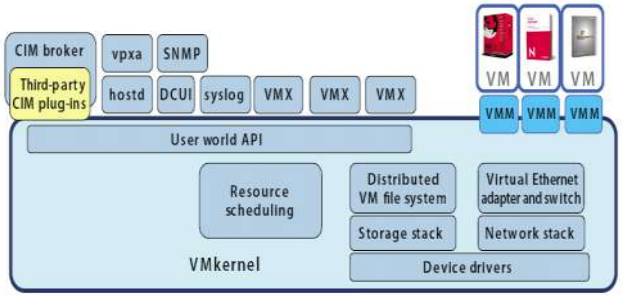

Fig.2.VMware ESXi architecture

V. HYPOTHESIS OF EXPECTED BEHAVIOUR

Since we are using a Type-1 hypervisor that works directly on hardware, the total processing time for each workload T_W can be described by the following equation:

$$T_W = T_{RR} + T_{RS} + T_{WR} + T_{WS} \quad (1)$$

where T_{RS} and T_{RR} represent sequential and random read time respectively, while T_{WR} and T_{WS} represent random and sequential write time respectively. For every specific workload we have an expected access time for the file system which includes five components as shown in following equation:

$$T_{WORKLOAD} = T_D + T_M + T_{FL} + T_{FB} + T_J + T_{HK} \quad (2)$$

where $T_{WORKLOAD}$ represents the overall time for finishing all operations on the current workload, and T_D, T_M, T_{FL}, T_{FB}, T_J, T_{HK} represent time needed for completing all operations related to directory, metadata, free list, file block, journaling and house-keeping operations in the file system, respectively.

In this study we have a specific situation where there are two sides which have identical settings of the operating system (CentOS) and the file system (XFS), used in the performance testing: (1) Native machine (hostOS) and (2) ESXi + VMs(guestOS).

1. Native hostOS: The time to process the generated workload depends on the benchmark interaction with the hostOS file system and also the characteristic of the file system. Total time to process the workload, $T_{W(native)}$ is defined as:

$$T_{W(native)} = f(benchmark, hostOS\_FS) \quad (3)$$

2. ESXi + VMs(guestOS): The time to process the generated workload ($T_{W(ESXi)}$) in this case depends on the benchmark interaction with guestOS file system, the characteristic of the file system and the virtualization processing component of the ESXi hypervisor (ESXi_proc) is as in the following formula:

$$T_{W(ESXi)} = f(benchmark, guestOS\_FS, ESXi\_proc) \quad (4)$$

Since we use the same settings as the native machine for our virtual machines, the benchmark interaction and characteristics of the file system on the guest will be the same as the ones on the native machine. The virtualization processing component depends on the virtualization type and hypervisor processing as in the following formula:

$$ESXi\_proc = f(virt\_type, hyp\_proc) \quad (5)$$

In the context of virtualization type, ESXi uses full virtualization, which is further enhanced with one of the technologies (depending on hosts' CPU) for hardware assisted virtualization. In the context of the hypervisor processing it is important to consider the delay, which represents the time required for the hypervisor to receive requests from virtual hardware of a guest OS and forward them to the hosts' hardware for proccessing. The delay can be explained as following: virtual machines generate workload, which passes from a VM through the hypervisor onto the hosts' hardware. First the benchmark application generates the workload which is passed on for further processing to the hypervisor. The second part happens inside the hypervisor and is defined as the interaction between guest workload and VM image file. Generated workload is passed on the hypervisor, which maps it into requests for VM large image files. Lastly the hypervisor's mapping process generates input files as requests for real disk drivers on the hosts' hardware. The time needed for generating those requests depends on the hypervisor's file system and caching capabilities.

The expected outcome according to formula (3) is that the native host will perform better than ours ESXi virtual machines. Virtual machines have a complex data path, formula (5), where data must pass through guest OS file system and the hypervisor onto machine hardware. Therefore, it is expected

that a degradation of the ESXi VM performance will happen compared to the native host machine, formula (4).

We have investigated a few cases for the this paper: firstly, the performace of a native host machine, then the performance of a single ESXi VM running and lastly the performance of several virtual machine running at the same time. In general, we expect:

- Native host to perform better when compared to ESXi with one virtual machine running.

- Running several instances of the ESXi virtual machines, *n**ESXi VMs (*n=1,2,3…*), should have a significant performance degradation compared to the native host.

VI. Test environment configuration

The assumption of an adequate testing is the application of a single hardware configuration, the same operating system, and measurement methodology for all test procedures as mentioned before. The hardware configuration contains all the components necessary for a modern-day computer, and in this case, it is a home-based system of the newer generation (Table 1). CentOS version 7.7 is selected as the operating system, which is currently one of the most popular Linux distribution.

During the installation process we opted for Gnome graphical interface installation option with essential packages and programs for a graphical environment. The XFS file system characteristics and layouts are shown in Table 2. Filebench is a program designed to measure the performance of file systems and storages, and it is capable of generating multiple workload types that simulate environments when using certain servers/services such as mail, web, file, database, etc. Before starting tests, we made sure that all available updates were installed. Each virtual machine was given 4 GB of RAM.

TABLE I
Hardware configuration of the test PC

MB	Gigabyte B75M-D2V
RAM	DDR3 1330 MHz, 16 GB
CPU	Intel
Model	Pentium G860
Cores	2 /2 threads
Speed	3.00 GHz
Cache(L1,L2,L3)	2x32kB; 2x256kB, 3MB
SSD	Samsung SSD 860 EVO
Interface	SATA 6Gbps
Capacity	250 GB
OS	CentOS 7.7.1908.el7

This benchmark behaviour is controlled using files with the extension *.f that are written in Workload Model Language, that can be edited in any text editor. The use for individual measurements involves putting a command from a terminal with root privileges using the name of the *.f file as an argument.

TABLE II
Fs layout

File system	Size	Mount
/dev/mapper/data-root	35 GB	/
/dev/sda1	4 GB	swap
/dev/sda2	1024 MB	*/ boot*

VII. Tests and results

The focus of this paper was to measure the performance of hard disks and data-flow in one of the more popular virtualization systems, especially in cases where several instances of virtual machines are being used. The main idea was: as the number of instances increases, there is a significant drop in performance and this drop is constant on any hardware-software configuration. Benchmark of the host computer without virtualization was taken as a reference point for file system performance in these tests.

A number of modified files of the source code ***fileserver.f***, ***webserver. f***, ***randomfileaccess.f*** and ***varmail.f*** were used during the tests, which are thus testing the files, web and the mail server environments, respectively. The changes were taken into consideration when setting the benchmark parameters in a way to provide as realistic as possible exploitation conditions. And while the location (/ bench), the I/O block size (iosize = 1M) and the average size of the add-on (meanappendsize = 16k) are common denominator for all tests, the parameters such as the number of files (nfiles), the average depth of the directory (meandirwidth) the average file size (meanfilesize), cache and the number of threads (nthreads) are changed on a case-by-case basis (with * .f files). The defined settings are retained throughout the entire benchmark test and are displayed in Table 3. For an easier view in the following table the name of each benchmark workload has been abbreviated with their initials (file server (FS), web server (WS), mail server (VMail) and random file access (RFA).

TABLE III
Settings of the source code in the *.f files

	FS	WS	VMail	RFA
nfiles	10.000	1.000	1.000	10.000
meandirwidth	20	20	1.000.000	20
meanfilesize	16k	16k	16k	
nthreads	50	100	16	5
cached				false

The duration of each test was 120 seconds, which is also stated in the *.f files, with the goal of acquiring the most realistic results. Special attention was paid to keep the OS clean and the impact of any external subject on system components was reduced to the minimum. After performing a reference measurement of the host computer without virtualization, ESXi

was installed and three virtual machines were generated. Tests were conducted in a way that one virtual machine was first started and measured, then two and three machines simultaneously. From the generated data, the final conclusions were made by calculating the average values of the results.

TABLE IV
BENCHMARK RESULTS (MB/S)

	FS	WS	VMail	RFA
Host	401.6	127.9	51.4	7379.5
1VM	230.4	67.6	45.7	3646.0
2VM	122.3	42.8	24.4	2126.9
3VM	78.2	24.9	14.8	1498.6

Table 4 shows the data we collected from workloads running in the test environment (again we used the same abbreviations like in Table 3). Data from Table 4 are shown on the next few figures, with remarks on the performance displayed in each. All of the measures shown in the following figures are displayed in megabytes per second (MB/s).

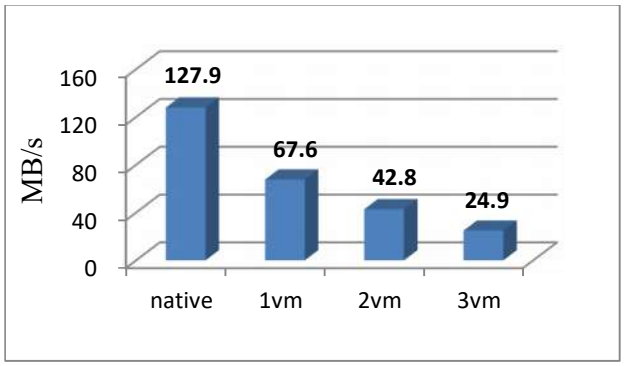

Fig. 3. Webserver.f workload test results

The characteristics of the webserver.f workload with our specification (100 threads) is that random reads dominate, there are some random write components, while the sequential components are not present. Here we observe that native host OS performs much better than in the case with one instance of the ESXi virtual machine (Figure 3). In the case of this workload, instantiation of more than one ESXi virtual machine brings some performance degradation but not significant.

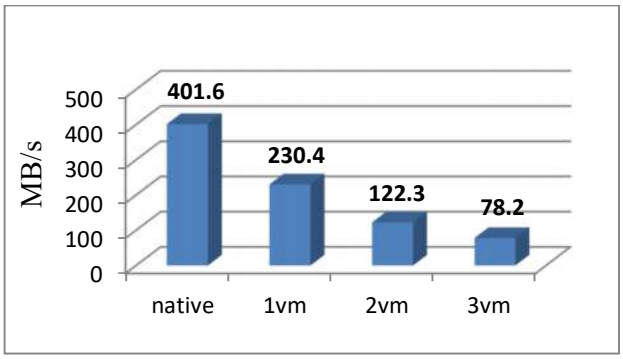

Fig. 4. Fileserver.f workload test results

The characteristics of the fileserver.f workload with fifty threads, are that both random and the sequential components dominate, but there is also a large number of I/O requests and much heavier data flow. A general notion is that, in the case for one virtual machine instance, the ESXi is significantly weaker than in the case of the native host OS. In the case of fileserver.f workload, when two virtual machines are instanced, the performance is further degraded by approximately the same amount as in the previous case (with one virtual machine). Instancing a third virtual machine brings very little performance degradation (Figure 4).

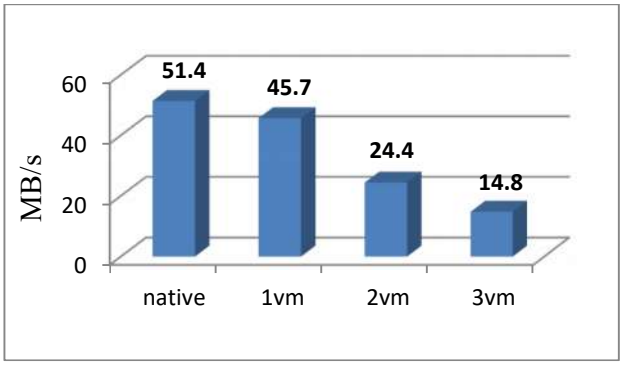

Fig. 5. Varmail.f workload test results

The characteristics of the varmail.f workload with our specification (16 threads) are that the components of random read and write are dominating, while the sequential components are not present, as it is shown in Figure 5. The special characteristic is that the components of the random write are synchronous, so each write will end up on the disk. The general notion for one instance is that performances of ESXi virtual machine are close to the native host OS. However, synchronous entries cancel the effects of cashing, so there are minor differences between native host OS and one instance of ESXi virtual machine. In the case of varmail.f workload, the instantiation of more than one ESXi virtual machines does not bring significant performance degradation.

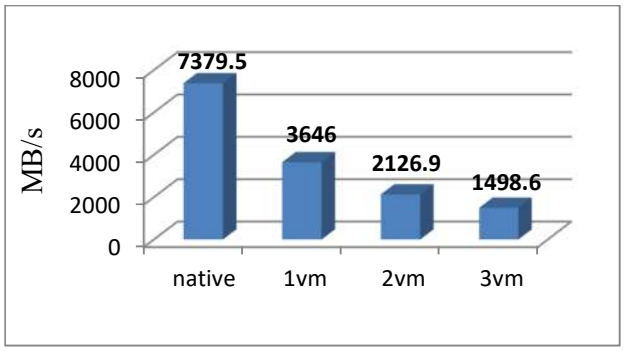

Fig. 6. Randomfileaccess.f workload test results

The characteristics of the randomfileaccess.f workload with five threads are that random reads dominate, while the sequential components are not present as shown on figure 6. We set up this workload so that cache would not be used. As we observe, the native host OS performs significantly better compared to one instance of ESXi virtual machine running. After starting second and third instances of ESXi virtual machines, we were able to observe that performance degradation is still present but it is not too significant.

The acquired benchmark results are fully expected and in line with the theoretical assumptions. The ESXi hypervisor and the hardware assisted full virtualization model show clear limitations on the data flow, in particular with the increase in the number of active virtual machines that cause even greater sharing of processors' resources and its increased use for hardware simulation. The addition of new instances of virtual machines is even more decreasing the achieved data flow, which means that new virtual machines cannot be added to the indefinite, as the performance of the whole system degrades per virtual machine added.

VIII. CONCLUSION

The introduction of virtualization has led to major changes in the use and deployment of information technology. Virtualization technology has a significant impact on reducing hardware investment as well as reducing operating costs, while also providing many additional benefits other than server consolidation. The great expansion of cloud computing in recent years has also contributed to the accelerated development of virtualization technologies and in the foreseeable future virtualization will always have an increased application in information technologies. It is also reasonable to expect, given the development of information technology today, that virtualization techniques will continue to improve and that the performance gap between virtualized systems and native systems will narrow in the future.

The results of our measurements showed that a native machine works convincingly better in most cases than a virtual machine based on ESXi hypervisor and full virtualization, as we assumed in our hypothesis of expected behaviour. Virtual machines running on the ESXi hypervisor have lower performance than a native machine, and in three of the four tests the performance degradation is approximately 50% when we have only one instance of a virtual machine running. The performance degradation is even greater with the introduction of more virtual machines. In one of the tests (mail server), the performance degradation between a native and a single virtual machine is not large, but with the introduction of new virtual machines into the test environment, the performance degradation of virtual machines becomes extremely pronounced. Future research may include a different approach where instead of comparing native machine vs. virtualized one, we compare different types of similarly structured virtualized machines or systems. With this research, we have proven that virtual systems still cannot reach the performance of non-virtualized systems, but as technologies evolve at an accelerated pace, we hope that in the future the performance of virtual machines will reach or be equal to regular non-virtualized machines.

ACKNOWLEDGEMENT

The work presented in this paper has partially been funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

LITERATURE

[1] C. Jiang, B. Luo, J. Wang, J. Zhang, Y. Wang, W. Shi, "Energy efficiency comparison of hypervisor," Proc. 2016 Seventh International Green and Sustainable Computing Conference *(IGSC)*, pp. 1-8, 2016.

[2] Correia, "Hypervisor based server virtualization" in *Encyclopaedia of information Science and Technology*, IGI Global, 2015, pp. 1182-1187.

[3] W. Huang, J. Liu, B. Abali, D. K. Panda, "A case for high performance computing with virtual machines," Proc. of the International Conference on Supercomputing. ACM, pp. 125-134, 2006.

[4] G. Casale, S. Kraft, D. Krishnamurthy, "A model of storage I/O performance interference in virtualized systems," Proc. of 31st International Conference on Distributed Computing Systems Workshops, pp. 34-39, 2011.

[5] J. Che, Q. He, Q. Gao, D. Huang, "Performance Measuring and Comparing of Virtual Machine Monitors," Proc. of 5th International Conf. Embedded and Ubiquitous Computing. EUC2008, Vol. 2, Piscataway, NJ, USA, pp. 381–386, 2008.

[6] A. Bhatia and G. Bhattal, "A comparative study of various hypervisors performance," International Journal of Scientific and Engineering Research, vol. 7, no. 12, pp. 65-71, 2016.

[7] J. Hwang, S. Zeng, F. Y. Wu, and T. Wood, "A component-based performance comparison of four hypervisors," Proc. of 2013 IFIP/IEEE International Symposium. IEEE, pp. 269-276, 2013.

[8] Pousa, Duarte; Rufino, José, "Evaluation of type-1 hypervisors on desktop-class virtualization hosts," IADIS International Journal on Computer Science and Information Systems. ISSN 1646-3692. 12:2, p. 86-101, 2017.

[9] H. Kazan, L. Perneel, M. Timmermann, "Benchmarking the performance of Microsoft Hyper-V server, VMWare ESXi and Xen hypervisors," Journal of Emerging Trends in Computing and Information Sciences, vol. 4, no. 12, pp. 922-933, 2013.

[10] Filebench project, Available: https://github.com/filebench/filebench/wiki

[11] VMware vSphere Documentation: https://docs.vmware.com/en/VMware-vSphere/index.html

[12] VMware ESXi 6.7 project: https://docs.vmware.com/en/VMware-vSphere/6.7/vsphere-esxi-vcenter-server-67-storage-guide.pdf.pdf

[13] VMware, Inc. white paper. "Virtualization Overview". https://www.vmware.com/pdf/virtualization_considerations.pdf

[14] VMware, Inc. white paper. "The Architecture of VMware ESXi," p. 3, 2020. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/ESXi_architecture.pdf