CASE STUDY OF PERFORMANCE ON THE KVM HYPERVISOR-BASED VIRTUALIZATION RELATED TO NATIVE HOST

Borislav Đorđević*,
[0000-0002-6145-4490]
Kristina Janjić,
[0009-0005-0750-6105]
Nenad Kraljević
[0009-0008-7684-5444]

School of Electrical and Computer Engineering,
Belgrade, Serbia

Abstract:
This study examines how file system performance differs between a native operating system and a KVM hypervisor-based virtualized environment. The research uses CentOS 9 as both the native/guest OS and employs Filebench for benchmarking purposes. Tests were conducted on both the native OS and within KVM virtual environments configured including one, two, three, and four virtual machines. The study establishes a mathematical model to compare performance between the native and virtual environments. According to the model, the native operating system exhibits significantly faster performance compared to its virtualized counterpart. Empirical results of the paper confirm this prediction, demonstrating a noticeable drop in file system performance.

Keywords:
KVM, Virtualization, Native OS, Linux, Centos 9, Virtual Machine.

INTRODUCTION

Virtualization technologies represent pivotal advancements in Information Technology and Cloud Computing. Hypervisor-based virtualization, the predominant form, enables running multiple full operating systems simultaneously on a single physical machine, facilitated through virtual machines (VMs). There are several types of hypervisor virtualization, including full hardware virtualization, paravirtualization, and operating system-level virtualization. Among these, full hardware virtualization is the most widely adopted. Virtualization offers numerous benefits over traditional single-operating-system architectures. One of the primary advantages is enhanced CPU utilization on physical servers. By enabling the simultaneous execution of multiple operating systems, virtualization optimizes hardware utilization and reduces energy consumption [1]. This capability is fundamental in modern computing environments, facilitating efficient resource allocation and scalability in IT infrastructures and cloud services.

Correspondence:
Borislav Đorđević

e-mail:
borislav.djordjevic@viser.edu.rs

DOI: 10.15308/Sinteza-2025-89-96

Virtualization indeed brings numerous advantages, but one significant drawback is the noticeable performance decrease for virtual operating systems compared to native environments. In a traditional architecture, a native operating system operates straight on the underlying hardware, achieving optimal performance. However, when the same operating system runs within a virtualized environment, it accesses hardware resources through the hypervisor and the host operating system, leading to a significant performance reduction. This performance degradation is inherent in virtualization scenarios regardless of whether a single virtual machine or multiple virtual machines are running concurrently. The additional layer of abstraction introduced by the hypervisor and the sharing of physical resources among VMs inevitably result in decreased performance compared to running on dedicated physical hardware.

2. RESEARCH OBJECTIVES, MOTIVATION, AND GOALS

This paper focuses on comparing FS performance between hypervisor-based virtualization and native operating systems running on physical machines. In related studies, numerous papers have addressed similar issues, investigating the performance of different hypervisors, including VMware ESXi, KVM, Xen, Proxmox, and Hyper-V. Such studies rely on diverse hardware configurations and make use of established benchmarking tools like Filebench, Fio, Bonnie++, Postmark, HD TunePro, Iozone, and LMbench [2-8]. A majority of sources in the literature do not employ a mathematical model; instead, they conduct high-quality experiments on specific hardware setups. The findings from these experiments are highly applicable in practical scenarios.

Several papers have investigated comparisons between native operating systems and virtualized operating systems [7-14]. While a minority of studies suggest minimal performance degradation [7-8], the majority indicate significant drops in performance, aligning with our findings as presented in our paper. Our approach shares similarities with references [9-14] but distinguishes itself through an expanded mathematical model and extensive experiments. These experiments include additional workloads, a larger number of virtual machines (VMs), and unique results across diverse hardware configurations.

The primary goal of this paper is to assess the FS efficiency differences between a native operating system and a virtualized operating system utilizing the KVM hypervisor (type-1). The study uses identical hardware configurations for both the native host operating system and the hypervisor with its configured virtual machines. Filebench is employed as the tool for performance measurement [15]. Central to this research is a foundational mathematical model, pivotal in interpreting the test results. This model is designed to be scalable, versatile, and applicable to similar case studies. The main focus lies in evaluating how virtualization impacts FS performance, with a specific emphasis on comparing the performance of the native operating system against that of the hypervisor running a single virtual machine. Additionally, the experiment is expanded by increasing the number of virtual machines from 1 to 4. The hypothesis that the native operating system outperforms KVM virtual machines was confirmed by the experiment's findings. This research contributes significantly by providing insights into the performance degradation associated with virtualization, supported by a robust mathematical framework.

3. KVM

KVM (Kernel-based Virtual Machine) technology holds a crucial role in Linux-based virtualization. Originally introduced with backing from Red Hat, KVM has been an integral part of Linux's core functionality starting from version 2.6.20. It operates as a kernel module, blurring the distinction between type-1 and type-2 hypervisors. KVM extends the capabilities of Linux's core architecture by enabling it to function as a native hypervisor. This setup allows Linux to host virtual machines directly, leveraging its own capabilities. Unlike standalone hypervisors, KVM doesn't require additional software like QEMU to manage VMs; instead, it utilizes existing Linux functionalities. As a hosted hypervisor, KVM leverages Linux as both the host operating system and the hypervisor, effectively embedding virtualization features within the Linux framework. In this hosted hypervisor setup, KVM utilizes QEMU [16] hardware virtualization without relying on its own.

4. MATHEMATICAL MODEL AND HYPOTHESES REGARDING EXPECTED BEHAVIOUR

In this paper, we look into the performance relation between native host operating systems and identical operating systems deployed through hypervisors. Our objective is to establish a model that evaluates FS performance across physical architectures and hypervisor

environments. In evaluating FS performance, benchmark tools are employed to generate specific workloads. These benchmarks typically encompass four main types of cycles: random reading, random writing, sequential reading, and sequential writing. Additionally, write operations can be categorized as synchronous or asynchronous, reflecting the significant influence of file system caching on write performance. Each benchmark workload simulates a diverse range of operations within the file system. These operations include managing directories, handling metadata, managing free lists, manipulating file blocks, and performing various housekeeping and journaling operations.

For modeling traditional architecture, we noticed there are the physical hardware and the host operating system (with the kernel and the file systems). The model involves three objects: a benchmark, a host kernel, and a host file system, so it means that the whole data path is simple. In the data-path, the benchmark produced the requests to the host kernel, and then the kernel forwarded these requests to the host file system. Workload processing time, TW_{hostOS}, depends on the benchmark, host kernel, and host file system, Equation 1:

$$TW_{hostOS} = f(Bch, hFS)$$

Equation 1. Host OS workload processing time

In Equation 1, ***Bch*** is the benchmark processing, including the file sets with their own file set operations. On the basis of the file set operations, the benchmark produces the requests to the host kernel. The kernel processes the requests, and then the kernel produces the requests to the host file system; this processing in the host file system is denoted as the ***hFS***. The processing of the host file system is highly complex, involving the characteristics of three objects: the host file system, the file system cache, and the physical disk drivers.

For hypervisor virtualization, the workload processing time is much more complex, while the data path in virtualization depends on the large number of components. These are the features of the following: file system types on both the guest and host sides, the guest and host file system caching, the virtual machine image file, hypervisor processing, and the hypervisor parameters. Data-path includes seven objects: benchmark, guest kernel, guest file system, virtual hardware, hypervisor, host kernel, and host file system. The model includes the three kernels, and two file systems with their own file system caches in the form of the FS-pair, and so it is a solidly complex data path, Equation 2.

$$TW_{hyp} = f(B, gFS, VH_{proc}, Hyp_{proc,}, hFS)$$

Equation 2. Hypervisor virtualization workload processing time

The first component, ***B*** denotes the benchmark requests for the guest OS file system. The second component, ***gFS***, marks the processing of the guest OS file system, correlated with the kernel of the guest OS. This component is quite similar to the 5th component, ***hFS***, and these components support different file system types. The third component, VH_{proc}, is the actual processing of the virtual disk drivers. The fourth component, Hyp_{proc}, refers to the hypervisor processing time, during which the hypervisor receives requests from the virtual disk drivers and forwards them to the virtual machine image file (VMI) in the host file system. The fifth component, ***hFS***, refers to the host OS file system time processing, which is closely integrated with the kernel of the host operating system. This component operates with a large VMI file. The second and the fifth components of Equation 2 are very correlated and must be considered an FS-pair, involving the complex relation of two FS caches.

Observing the data path in Equation 1 (three input factors) and Equation 2 (five input factors), the data path is far more complex in virtualization. In summary, while native host operating systems generally exhibit superior FS performance due to direct hardware access, significant performance degradation is expected in virtualized environments. This degradation can vary based on workload characteristics and a multitude of factors related to physical hardware, operating systems, and virtualization technologies.

5. TEST SETUP AND BENCHMARKING APPLICATION

We focus attention on a fair comparison of file system performance achieved through identical hardware, virtual machines, measurement methodologies, operating systems, and a standardized benchmarking program. We used the KVM virtual platforms: QEMU emulator version 8.1.2 (pve-qemu-kvm_8.1.2-4) on a Linux host OS: Debian 12 Bookworm, kernel: 6.5.11-4 / ext4, while the experiment was conducted on an HP server running CentOS Stream 9 as the native guest operating system. The HP server features the following configuration:

- **CPU**: Intel® Xeon® Silver 4116 CPU @ 2.10GHz

- **RAM**: 32GB DDR4 2400 MHz
- **Hard disk**: 2x HPE 480GB RAID1, SATA 3, Sequential read up to 535 MB/s, Sequential write up to 495 MB/s
- **Host Operating Systems**: KVM: QEMU version 8.1.2 on Debian 12 Bookworm, kernel: 6.5.11-4, ext4

Each experimental test utilized Filebench 1.4.9.1-3 as the benchmarking tool. Filebench facilitates the simulation of diverse server environments by defining various workloads, offering detailed performance metrics such as file read/write throughputs [16]. The storage setup comprised two identical hard drives configured in RAID-1, housed within the HPE ProLiant BL460 Gen10 server. Both native and virtual environments were tested, with virtual machines stored on the same RAID-1 disks. Below are the parameters used for the virtual machines:

- **Number of virtual CPU assigned to each VM**: 4, Virtual memory assigned to each VM: 8GB
- **Virtual hard disk assigned to each VM**: 64GB (/dev/sda), 32GB root FS, 32GB testing FS (XFS)
- **Guest OS**: CentOS Stream 9

All performance tests were conducted using Filebench, a well-established file system and storage benchmarking tool. Filebench is renowned for its ability to simulate a wide range of workloads that closely resemble real-world server environments. These workloads can mimic services such as mail servers, web servers, database servers, file servers, and more. This capability allows for comprehensive testing and evaluation of file system and storage performance under conditions that mirror practical server usage scenarios.

Mail server results for the native host OS and for the virtual machines are shown in Figure 1.

Features of the mail server workload are the following: dominant random reading and random writing (synchronous), without sequential components. Other features are a moderate dataflow and a moderate number of input/output requests. As a consequence of predominant random reading and synchronous random writing, the caches of both file systems have a small performance influence. For the KVM mail server workload, watching the native host operating performance related to a single virtual machine (1VM), we detected remarkable performance degradation (about 2.73 times). As the number of virtual machines (1VM-4VM) increases, we determined the further performance drops (about 27-67%). For the KVM mail server workload, all speeds are solidly lower than the max disk speed (500MB/s). It means that FS cache effects are almost zero. For the KVM mail server, the differences between native and virtual operating systems are a consequence of Equation 1 vs. Equation 2. For a small dataflow and predominant synchronous random writing, the FS caches have no impact, so these differences are remarkable. For virtualization, in the view of Equation 2 and KVM mail server, we consider that two components have the dominant performance influence: the VH_{proc} component (Equation 2) above all and then the Hyp_{proc} (Equation 2), while some small influence is due to the FS components (***gFS and hFS***) (Equation 2), which operate as an FS-pair, exclusively. For the KVM mail server workload, the impact of two file system caches (as part of an FS-pair) is insignificant; the main reason is the dominant synchronous random writing/random reading.

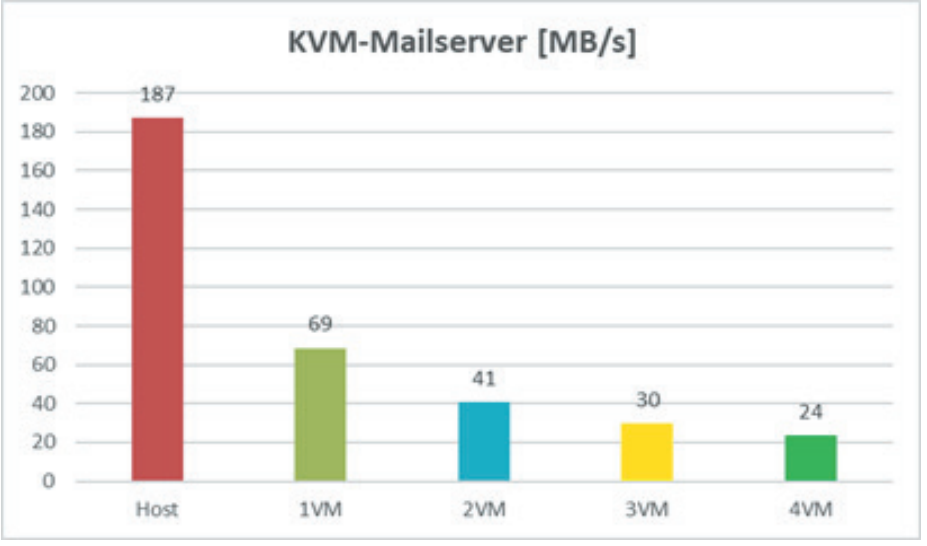

Figure 1. Mail server performance test results

Results for the web server for the native host OS and the virtual machines are shown in Figure 2.

Features of the web server workload are the following: a lot of random read components and small random write components (as the log-appending), whereas sequential transfers do not exist. Other features are the small dataflow and a moderate number of input/output operations. By the predominant random read components, the file system caches have inconsiderable performance influence unless the readings are with repetition. For the KVM web server workload, regarding the native host FS performance related to a single virtual machine, we noticed a relatively small performance drop (6%). With increasing the VM number (1VM-4VM), we detected the additional performance degradation (5%-26%). For the KVM web server workload, all speeds are lower than maximal disk speeds. Anyway, the speeds are high for random read workload, which means the repeated random reads exist, so the cache effect is noticed, but the random read data traffic for virtual/physical drivers is dominated, also. For the KVM web server, the speed differences between native and virtual operating systems are a consequence of Equation 1 vs. Equation 2. For a small dataflow with repeating random reading, the FS caches relieve these differences remarkably. Seen through the virtualization, in the view of Equation 2 and KVM web server, we consider that two components have the most performance impact. These are the Hy_{pproc} (Equation 2) and the VH_{proc} (Equation 2), whereas the certain influence is by the FS components ***gFS***, and ***hFS*** (Equation 2), in the form of the FS-pair. For the KVM web server workload, the influence of two file system caches (in an FS-pair) is solid; the reason is the dominant repeated random reading.

Fileserver results for the native host OS and the virtual machines are shown in Figure 3.

Features of the fileserver workload are the following: the dominant random/sequential reading and random and random/sequential write components; in other words, all kinds of transfers are present. Other features are the large dataflow and the large number of input/output requests. By the repeated reading and lots of writing, the file system caches have a solid performance influence. For fileserver workload, observing the native FS performance related to a single virtual machine, we noticed large performance drops (2.72 times). By increasing the number of VMs (1VM-4VM), we detected the additional performance degradation (11%-36%). For the KVM fileserver workload, the speeds are higher than the maximal disk speed (500MB/s) for all virtual machines. Anyway, the speeds are high for this kind of workload. It means that the FS cache impact is large for whole writing and for repeated reading, but the disk traffic for virtual/physical drivers is significant, also. For the KVM fileserver, all differences between native and virtual operating systems are a consequence of Equation 1 vs. Equation 2. Despite the large cache effects, the differences are still large. Seen through the virtualization, in the view of Equation 2 and the KVM fileserver, we identify two components as having the most significant impact on performance influence, the VH_{proc} and Hyp_{proc} components. The big influence is due to the components ***gFS*** and ***hFS*** (Equation 2), which operate as an FS-pair also. For the KVM fileserver workload, the influence of two file system caches is very remarkable; the main reason is the multiple repeated readings and a solid amount of writing.

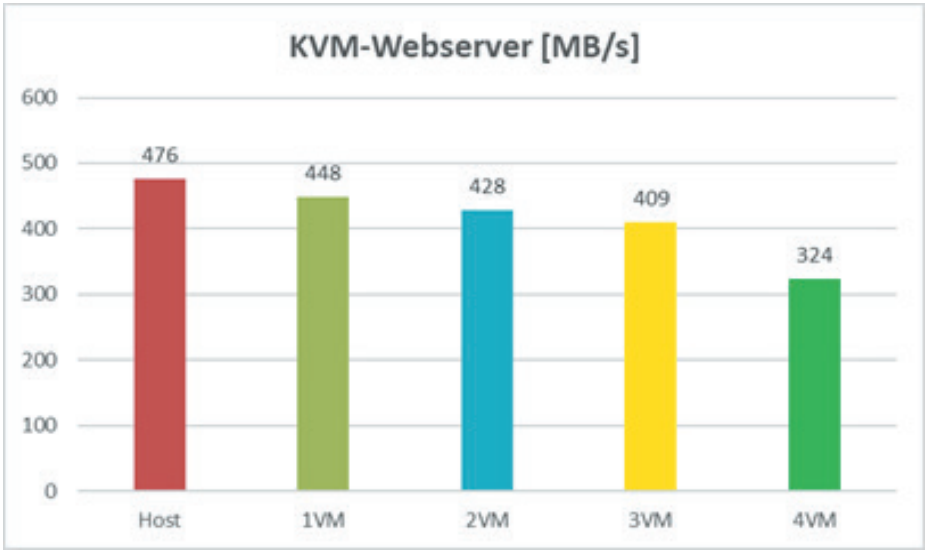

Figure 2. Web server performance test results

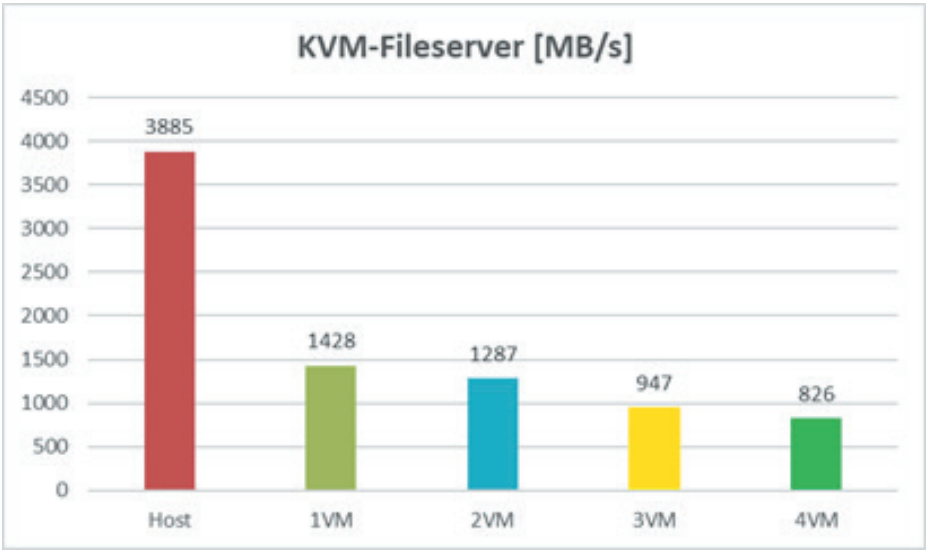

Figure 3. Fileserver performance test results

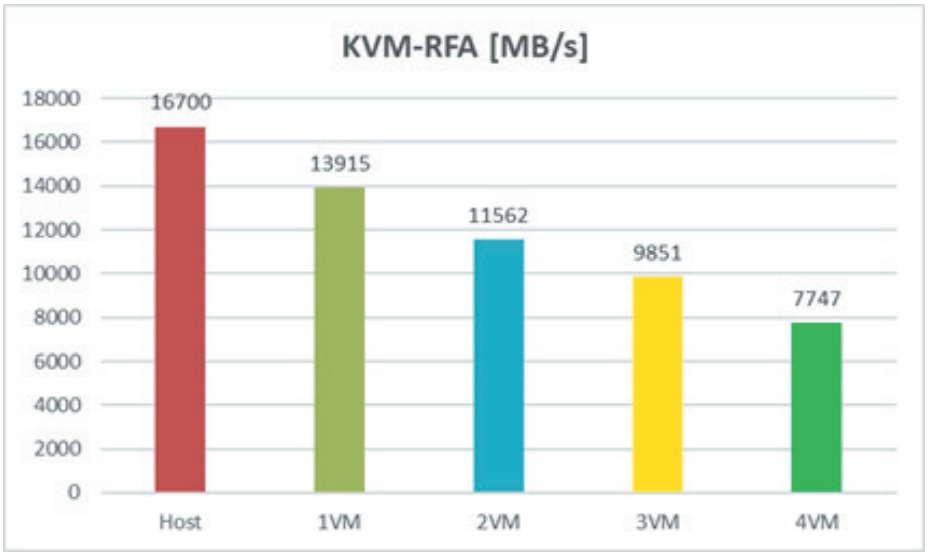

Figure 4. RFA performance test results

RFA results for the native host OS and the virtual machines are shown in Figure 4.

Features of the RFA workload are the following: the dominant random read components as well as asynchronous random write components; sequential transfers are very little present. Other features are the large number of input/output requests and the moderate dataflow. As a consequence of the dominant random asynchronous writes, the file system caches have a huge performance influence. For the KVM RFA workload, in the context of the native host performance versus a single virtual machine, we measured the small performance drop (20%). With increasing the number of virtual machines (1VM-4VM), we noticed the additional performance drops (17-27%). For the KVM RFA workload, all RFA speeds are solidly higher than the max disk speeds; it means that the FS cache impact is extremely large. For the KVM RFA, the differences between native and virtual operating systems are due to Equation 1 vs. Equation 2. However, for a repeating random reading and asynchronous random writing, the FS caches reduced these differences. Seen through the virtualization, in the view of Equation 2 and KVM, we consider that two components have the most performance influence, the Hyp_{proc} and VH_{proc} components (Equation 2). For the KVM RFA workload, the influence of two file system caches is huge; the main reason is the prominent asynchronous random writing.

When we collect all results together, we can see the overall picture of this case study, as shown in Table 1.

Table 1. Overall results

%	Host/1VM	1VM/2VM	2VM/3VM	3VM/4VM
Mail server	2.73 times	67%	37%	27%
Web server	6%	5%	5%	26%
FS	2.72 times	11%	36%	15%
RFA	20%	20%	17%	27%

For this experiment, watching the native host FS performance related to a single virtual machine, we noticed solid drops in file system efficiency for all or most workloads. We detected the most pronounced drop (2.7 times) for the most complex workload, fileserver (with large dataflow), and 2.7 times for mail server (with smaller data flow). However, for less demanding workloads (web server and RFA) we have detected a smaller drop, 20% for RFA and 6% for web server. With increasing the number of virtual machines (1VM-4VM), solid drops are for fileserver 10-36% and for mail server 27%-67%. Then, we detected relatively smaller drops: for web server 5-26% and for RFA 17-27%. These relatively small drops with the increasing number of virtual machines are a consequence of FS caches for a strong physical server and a relatively large amount of RAM allocated to virtual machines.

6. CONCLUSION

In this paper, if we are looking at the native host FS performance related to a single virtual, as the main evaluation parameter, we detected significant drops in file system efficiency, with the most pronounced drop being about 2.7 times. This drop was detected for the most complex workload, which contains a dominant data flow, and for mail server workload. Drops for web server 6% and for RFA 20% represent the smaller drops. With further increasing the number of virtual machines (1VM-4VM for our case), the FS performance drops can be relatively strong, for fileserver 10-36% and mail server 27%-67% workloads. For web server the performance drops continue (5-26%) and for RFA 17-27%, and these can be seen as the relatively smaller FS drops. Anyway, we consider that KVM hypervisor-based virtualization exposes the solid drops in file system efficiency.

For a comprehensive exploration of the FS performance relationship between native host operating systems and hypervisor-based virtualization, our mathematical model is highly adaptable. However, generating numerous case studies is essential.

These studies will contribute to building a Knowledge Data Base (KDB) focused on understanding FS performance degradation caused by virtualization. The case studies will encompass various aspects: different hardware configurations, various hypervisors, diverse operating systems and their kernels, different file systems, various benchmarks, etc. This effort represents a significant component of our future research agenda.

7. ACKNOWLEDGMENT

The paper has been funded by the Ministry of Education, Science, and Technological Development of the Republic of Serbia.

REFERENCES

[1] E. Correia, *Hypervisor based server virtualization*. Encyclopedia of Information Science and Technology, Third Edition, IGI Global, pp 1182-1187, doi:10.4018/978-1-4666-5888-2.ch112, 2015.

[2] M. Polenov, V. Guzik, V. Lukyanov, *Hypervisors comparison and their performance*, CSOC2018: Software Engineering and Algorithms in Intelligent Systems, Springer, pp148-157. doi;10.1007/978-3-319-91186-1, 2018.

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

[4] S. Pawar, S. Singh, "Performance comparison of VMWare and Xen hypervisor on guest OS", *IJICSE*, vol. 2, no. 3, pp. 56-60, 2015. https://ijicse.in/index.php/ijicse/article/view/43/41

[5] S. A. Algarni, M. R. Ikbal, R. Alroobaea, A. S. Ghiduk, F. Nadeem, "Performance evaluation of Xen, KVM, and Proxmox hypervisors", *Int. J. of Open Source Software and Processes,* vol. 9, no. 2, doi: 10.4018/IJOSSP.2018040103, 2018.

[6] P. Kedia, R. Nagpal, "Performance evaluation of virtual environment with respect to physical environment", *Int. J. of Computer Applications*, 89(11), pp 17-22, doi: 10.5120/15676-4425, 2014.

[7] V. K. Manik, D. Arora, "Performance comparison of commercial VMM: ESXi, XEN, HYPER-V & KVM", in *3rd Int. Conf. on Computing for Sustainable Global Development, New Delhi*, https://ieeexplore.ieee.org/document/7724572, 2016.

[8] A. Bhatia, G. Bhattal, "A comparative study of various hypervisors performance", *Int. J. of Sci, Eng. and Tech. Research*, vol. 7, no. 12, pp. 65-71, https://www.ijser.org/researchpaper/A-comparative-study-of-Various-Hypervisors-Performance.pdf, 2016.

[9] B. Đorđević, V. Timčenko, E. Nikolić, N. Davidović, "Comparing Performances of Native Host and Virtualization on ESXi hypervisor", IEEE in *20th INFOTEH-JAHORINA*, 21(1), pp 1-4, doi: 10.1109/INFOTEH51037.2021.9400648, 2021.

[10] B. Đorđević, S. Milenković, N. Davidović, V. Timčenko, "Performance comparison of native host vs. ESXi hypervisor-based virtualization", *8th IcETRAN*, https://www.etran.rs/2021/zbornik/Papers/101_RTI_2.2.pdf, 2021.

[11] B. DJORDJEVIC, V. TIMCENKO, N. KRALJEVIC, N. MACEK, *File System Performance Comparison in Full Hardware Virtualization with ESXi, KVM, Hyper-V and Xen Hypervisors*, Advances in Electrical and Computer Engineering (AECE), 21(1), pp 11-20, doi: 10.4316/AECE.2021.01002, 2021.

[12] B. Đorđević, V. Timčenko, D. Sakić, N. Davidović, "File system performance for type-1 hypervisors on the Xen and VMware ESXi", IEEE in *21st INFOTEH-JAHORINA*, doi: 10.1109/INFOTEH53737.2022.9751288, 2022.

[13] B. Đorđević, M. Marjanović, V. Timčenko, "Performance comparison of native host and hyper-based KVM virtualization", in *28th TELFOR*, doi: 10.1109/TELFOR51502.2020.9306550, 2020.

[14] B. Đorđević, M. Piljić, N. Kraljević, V. Timčenko, "Comparison of file system performance in full virtualization with MS Hyper-V and KVM hypervisors", in *9th IcETRAN*, https://www.etran.rs/2022/zbornik/ICETRAN-22_radovi/067-RTI2.5.pdf, 2022.

[15] Filebench - https://github.com/filebench/filebench. [Accessed 2023].

[16] H. D. Chirammal, A. Mukhedkar, A. Vettathu, *Mastering KVM Virtualization,* Packt Publishing Ltd, ISBN 9781784399054, 2016.