INFORMATION TECHNOLOGY SESSION

FILE SYSTEM PERFORMANCE COMPARISON WITH KVM AND XEN AS TYPE-1 LINUX-BASED HYPERVISORS

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 compares FS (file systen performance between bare-metal hypervisors built on the Linux architecture, focusing on KVM and Xen. While both divide Linux-based and QEMU-based architectures, detailed analysis reveals important differences. Filebench was chosen as the benchmarking tool for its adaptability in emulating real-world applications within authentic server environments. CentOS 9, a representative Linux distribution, served as the guest OS. Performance was assessed while running one, two, and three VM (virtual machine) concurrently, highlighting scalability under varying loads. The study introduces a mathematical model of the bare-metal virtualized environment to establish a theoretical framework for performance analysis. Empirical experiments complement the model, serving as specific case studies. By combining theoretical modeling with practical experimentation, the research provides a deeper understanding of the factors influencing FS performance in virtualized environments.

Keywords:

Hypervisor, KVM, Xen, CentOS 9, Virtual Machine - VM, Filebench.

INTRODUCTION

Virtualization is a transformative technology in contemporary IT, fundamentally altering the way resources are handled, stored, and utilized. By enabling several OS (operating system) to coexist on the same HW (hardware) platform, it enhances system reliability, availability, and resource efficiency. Virtualization enables the creation of VM, servers, along with other HW resources, leading to cost reductions, simplified system management, and greater expandability. Through precise allocation of HW resources like processing power (CPU), RAM, and disk space to VM, it ensures optimal HW utilization [1]. Although virtualization provides many advantages, it also introduces challenges, such as complex management, security vulnerabilities, potential software licensing expenses, and the vulnerability of having a single point of failure in case of hypervisor or physical server issues. However, the benefits of virtualization greatly outweigh its drawbacks, establishing it as a cornerstone of contemporary IT.

Correspondence:
Borislav Đorđević

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

DOI: 10.15308/Sinteza-2025-97-104

Virtualization includes several forms, including HW, SW (software), data, and desktop, as well as the virtualization of storage, memory, and network HW configurations. This study concentrates on HW virtualization, which allows VM to run as independent computers while sharing the same physical HW. This capability is enabled by the hypervisor, which creates and manages VM. Hypervisors abstract the HW from the OS, allowing numerous OS to operate in parallel on a single HW platform. Hypervisors are typically categorized into two types: type-1 hypervisors, or bare-metal hypervisors, running directly on the HW, and type-2 hypervisors, also known as hosted hypervisors, functioning within an existing OS environment.

2. MOTIVATION, RESEARCH OBJECTIVES, AND PURPOSE

A variety of scholarly articles within the virtualization field employ different techniques to evaluate the efficiency of different virtual environments. One widely adopted approach is to conduct comparative analyses that examine the performance of different hypervisors [2-10]. These analyses usually consist of meticulously planned experiments and rely on widely recognized benchmarks. However, it is noteworthy that a large number of these studies omit the use of mathematical modeling when evaluating virtual environments.

References [11-15] employ analogous mathematical models in the present paper, yet they concentrate on distinct hypervisors and operate on diverse HW specifications.

The primary purpose of this study is to develop a thorough mathematical model aimed at evaluating FS performance in virtual environments using type-1 hypervisors built on the Linux architecture. This model incorporates a broad set of input arguments and is made to be expandable for future improvements. What differentiates our approach from others is the methodology we apply: we start by developing a mathematical model, followed by an exploratory setup that acts as a particular practical analysis. This math model plays a pivotal role in analyzing the outcomes of our experiment, providing a distinct viewpoint on the assessment of the performance of virtual environments.

In this study, we focused on comparing KVM and Xen as type-1 hypervisors, based on the Linux framework hypervisors, utilizing QEMU-based full HW virtualization technology. The same HW conditions were applied for testing both hypervisors to ensure an impartial evaluation. The guest OS was CentOS 9, configured with the XFS FS. For the experiments, we employed the Filebench performance testing program, which includes different distinct workloads: fileserver, mail server, web server, and RFA (random-file-access). These workloads were selected to represent different types of typical virtualized environment tasks. Following the experiments, we validated our findings through the implementation of a mathematical model to analyze and clarify the results. This approach allowed us to derive meaningful insights into the performance characteristics of KVM and Xen across various workloads within the virtualized environment.

3. XEN AND KVM

The Xen platform consists of the Xen hypervisor, which runs directly on the physical HW, and multiple domains that function as VM operating atop the hypervisor. The main elements working together to provide efficient and flexible virtualization include:

- Xen Hypervisor: This is the core software layer that interacts directly with the HW. It is responsible for directing resources such as processing power, RAM, and input/output operations for multiple guest OS running simultaneously. The hypervisor ensures the parallel execution of these guest systems and supports various architectures, including x86, x86-64, Itanium, PowerPC, and ARM.
- Domain 0 (Dom0): Dom0 is a modified Linux kernel that is tightly integrated with the hypervisor. It holds exclusive rights for managing physical input/output resources and overseeing other VM (DomU guests). At startup, Dom0 is initiated and is tasked with both controlling the Xen hypervisor and running the device drivers for physical HW.
- Domain U (DomU): DomU refers to unprivileged guest systems that are not able to directly access physical HW. Instead, physical HW is managed by Dom0. These guests can be initiated through either a modified OS with paravirtualization or an OS that remains unchanged utilizing HW-assisted virtualization.

This hypervisor regulates access to the HW of the physical machine for guest domains. Instead of allowing guest domains to directly utilize privileged instructions, hypercalls are utilized to notify the Xen hypervisor of the need to execute privileged instructions, after which the hypervisor handles the request. The functionality of hypercalls is analogous to system calls in an OS. It serves as a software catch between the VM and Xen, much like how a system call acts as a software interrupt between an application and the kernel [15].

KVM (Kernel-based Virtual Machine) is a virtualization solution that is open-source and enables HW-level virtualization directly integrated into Linux and functions as part of its kernel (Figure 3). KVM was originally developed as a Red Hat-sponsored initiative and has seamlessly integrated itself into the Linux kernel starting from the version 2.6.20, functioning as a crucial module of kernel.

KVM utilizes parts of QEMU for emulating real devices. One of KVM's advantages is its ability to support a diverse range of OS for VM, including both Linux and Windows. The KVM hypervisor enables full virtualization, utilizing HW virtualization on supported processors to allow VM to operate without modifications to the guest OS. It offers every VM the full range of services typically found in a physical system, including a virtual-BIOS and HW such as a processor, memory, storage, network cards, etc. As a result, each VM fully simulates a physical computer [16].

4. MATHEMATICAL MODEL AND ASSUMPTIONS ON EXPECTED BEHAVIOR

The time required to process workloads in a hypervisor-based virtual environment, ***Tw***, is determined by a minimum of five key elements that exert significant influence, Equation 1:

$$Tw = f\,(Bn, gFS, VH\text{-}proc, Hyp\text{-}proc, hFS)$$

Equation 1. Wokload time in virtual environment

The initial element, ***Bn***, depicts the time taken by the benchmark to process data. The second element, ***gFS***, denotes the processing time of the FS within the guest OS. The guest FS is closely linked with several components, including the guest OS kernel and its file system cache mechanism. Both ***Bn, gFS*** expose comparable characteristics across all tested hypervisors. This similarity stems from the consistent use of the same benchmark features, parameters, VM configurations, and the guest FS (XFS).

VH-proc, the third element, depicts the processing time for virtual HW, primarily the virtual disk drivers. During our experiment, both KVM and Xen hypervisors show considerable differences in the ***VH-proc*** component. KVM exclusively employs full HW virtualization, while Xen supports both full HW virtualization and paravirtualization, although paravirtualization is not part of the scope of our experiment. It is important to note that both hypervisors rely on QEMU-based open-source technology for full HW virtualization. A key distinction, however, lies in the sets of virtual drivers provided by QEMU, which have evolved over a long period (from the release of qemu-0.10.0 on March 4, 2009, to the release of qemu-8.2.1 on January 29, 2024). Xen and KVM each incorporate different sets of QEMU virtual drivers, which significantly impact the ***VH-proc*** component. As a result, the performance characteristics of each hypervisor are expected to differ substantially. ***VH-proc*** is intricately linked with FS caching on both the guest/host OS sides. The differences in ***VH-proc*** between Xen and KVM can therefore impact how effectively each hypervisor manages FS caching, further influencing overall system performance in our experimental setup.

Hyp-proc, the fourth element, depicts the time allocated for hypervisor processing. This includes the time the hypervisor spends handling requests from virtual drivers and forwarding them to the host OS. In practical terms, FS requests from the guest FS are passed on to the host FS within the VMI (VM image file). In our experiment, we evaluated two distinct hypervisors: XenServer with the original Xen hypervisor, which follows the traditional Xen architecture, and KVM hypervisors, which use real Linux kernels with KVM kernel modules. These hypervisors are designed with a micro-kernel architecture, which is both lightweight and modern. Despite both hypervisors employing a micro-kernel design, they are expected to exhibit different hypervisor processing times due to their unique design philosophies and implementation details. These differences in hypervisor processing times (***Hyp-proc***) are crucial as they directly impact the overall performance and responsiveness of virtualized environments. Understanding these nuances is essential for accurately interpreting and comparing the performance results obtained from experiments involving Xen and KVM hypervisors.

hFS, the fifth element, depicts the processing time of the host FS, and we expect notable variations among hypervisors. In our testing environment, both hypervisors, being Linux-based, use two widely used FS options: ext4 and XFS, which may or may not involve the use of the LVM (Logical Volume Manager). For our setup, we selected ext4 for both hypervisors, though it's important to mention that the versions of ext4 used were not identical, which leads to inherent differences in performance.

When analyzing the host OS as a key element of the virtual environment, several important differences can be observed between the two Linux-based hypervisors, Xen and KVM. Both hypervisors operate with a distinct Linux distribution: XenServer utilizes customized Linux version tailored for Xen, while KVM can work with any Linux distribution. As a result, the host OS differs in several aspects, such as versions of physical drivers, host kernels, versions of host FS, and OS-system/graphical environments. While both hypervisors are Linux-based, they exhibit considerable variations in their host OS configurations and FS.

In summary, the features regarding the similarities and differences between the two hypervisors are as follows:

On the guest side, all features of the VM and benchmark remain the same.

On the virtual environment side, both hypervisors utilize full HW virtualization and are based on QEMU solutions for virtual drivers. However, significant differences arise due to the third (***VH-proc***) and fourth (***Hyp-proc***) components in Equation 1.

On the host OS side, despite both hypervisors being Linux-based, there are substantial differences in kernel versions, Linux distributions, host FS (both hypervisors use ext4, but with different versions), physical disk drivers, and FS cache mechanisms. In a virtual environment, an FS pair always exists (guest FS on host FS). Our experiment includes only one FS pair: XFS on ext4.

Using the mathematical model, along with the recognized similarities and distinctions between the hypervisors tested, we analyze and interpret the performance outcomes from the experiment.

5. TEST SETUP AND BENCHMARKING PROCESS

We emphasize fair and accurate performance evaluation by employing identical HW, VM, measurement techniques, OS, and a unified benchmarking tool. Fairness was maintained throughout the experiment by configuring consistent HW configurations, selecting s uniform OS for both the guest and host environments, and using a single benchmarking program across all stages of testing. The virtual environments employed were VMware ESXi version 8.0 and Xen Citrix Hypervisor version 8.2.1. The tests were conducted on an HP server running CentOS Stream 9 as the guest OS. The server's specifications are as follows: Intel® Xeon® Silver 4116 CPU @ 2.10GHz, 32GB DDR4 2400 MHz RAM, 2x HPE 480GB SATA 6G RI SSF SSDs in RAID1 configuration with sequential read speeds up to 535 MB/s and sequential write speeds up to 495 MB/s, running Xen Citrix Hypervisor 8.2.1 and KVM with 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).

All tests in the experiment were carried out using the Filebench 1.4.9.1-3 benchmark tool, which facilitates the reproduction of different real server scenarios by creating diverse workloads. It provides comprehensive performance data, including file read/write throughput for various types of workloads [17]. For storage, a pair of identical hard drives was set up in RAID-1 and mounted on the server (HPE ProLiant BL460 Gen10). Each virtual environment was tested with VM hosted on the same RAID-1 disks. The VM specifications are as follows: 4 virtual CPUs, 8GB of virtual memory, a 64GB virtual hard disk with 32GB allocated to /dev/sda1 (root FS) and 32GB to /dev/sda2 (testing FS with XFS), and the guest operating system is CentOS Stream 9.

6. EVALUATION AND FINDINGS

We single out that the primary goal of this study is to assess the FS performance of two dissimilar type-1 hypervisors using a range of workloads, including mail server, web server, fileserver, and random-file-access. Originally, performance was assessed with a single VM, followed by constant assessments with two, three, and four VM functioning concurrently.

The explanation of performance heavily relies on the features discussed in Chapter 4, including ***VH-proc, Hyp-proc***, guest FS, FS-pair, FS-cache-pair, physical and virtual disk drivers, and various elements of the host OS, such as the kernel, host FS, and OS/graphical environments. It's crucial to emphasize that many of these features differ significantly between Xen and KVM.

Figure 1 illustrates the outcomes of the fileserver workload test.

Figure 1. Results from the fileserver tests

The fileserver workload is defined by a rich mix of random and sequential read/write operations. This type of workload involves numerous input/output requests and a significant data throughput. FS caches play a crucial role, especially for frequent read operations and asynchronous writes. When examining the fileserver workload, KVM consistently outperforms Xen, with KVM being 2.37 to 3.92 times faster. The differences in FS performance are substantial. When observing the achieved throughput and the maximum disk speeds (around 500MB/s), both hypervisors surpass the maximum disk speeds with one VM. However, when using two VM, Xen's throughput drops below the maximum disk speeds, while KVM maintains higher throughput across all VM. These high throughputs indicate that FS cache pairs (guest/host caches) have a significant impact, while much of the I/O traffic also interacts with virtual-physical drivers due to the random and sequential nature of the workload. In the case of the fileserver workload, each element from Chapter 4 plays a role, but the most critical elements are ***VH-proc*** (including FS-pairs and drivers), disk drivers, and the cache effects of the FS-cache pair. Cache misses in both the guest and host caches make the virtual disk drivers of the guest OS and the physical disk drivers of the host OS very important factors. Given the substantial data throughput of the fileserver workload, it can be concluded that KVM benefits from a more efficient combination of components: ***VH-proc*** and the FS cache effects from the FS-cache pair, along with superior virtual/physical disk driver performance compared to Xen for random and sequential I/O requests.

Figure 2 illustrates the outcomes of the mail server workload test.

The mail server workload is mainly defined by a higher frequency of random reads and synchronous random writes, accompanied by a moderate volume of input/output operations and data throughput. Due to the prevalence of random reads and synchronous writes, the impact of FS caches on both the guest and host OS is anticipated to be minimal. For the mail server workload, Xen outperforms KVM, showing a performance improvement of 16-30-62%. When observing the achieved speeds and the maximum disk speeds, both Xen and KVM exhibit notably lower throughputs across all VM. The low mail server speeds suggest that the influence of guest/host FS caches is minimal, meaning most I/O traffic is directed to virtual/physical drivers. For the mail server workload, several components from Chapter 4 play a significant role. However, the primary components are ***Hyp-proc*** and ***VH-proc*** with FS-pair, although the FS cache effects are minimal in this case. Due to the limited influence of both FS caches, the virtual drivers of the guest OS and the physical disk drivers of the host OS become the most critical factors, particularly for random read/random write traffic. Given this, we conclude that Xen provides a more optimal combination of ***Hyp-proc***, ***VH-proc***, disk drivers, and minimal cache effects for the mail server workload.

Figure 3 illustrates the outcomes of the web server workload test.

The web server workload is defined by a prevalence of random reads and small random writes, accompanied by a fair volume of input/output requests and data throughput.

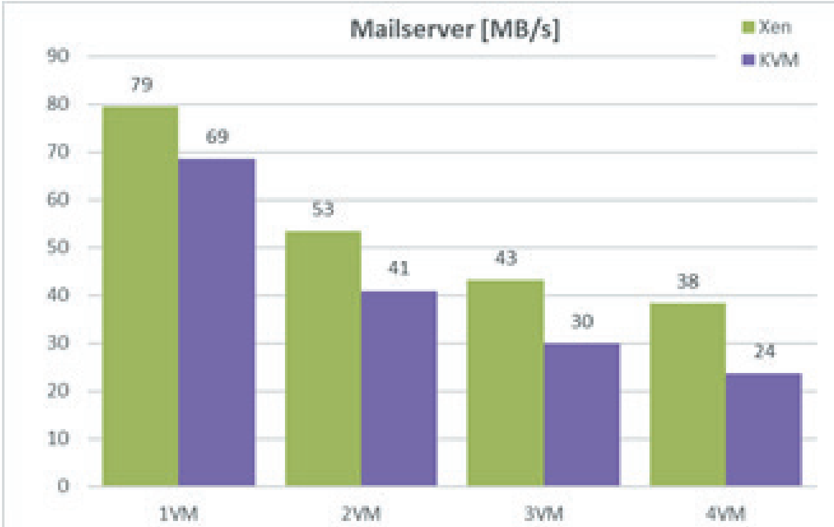

Figure 2. Results from the mail server tests

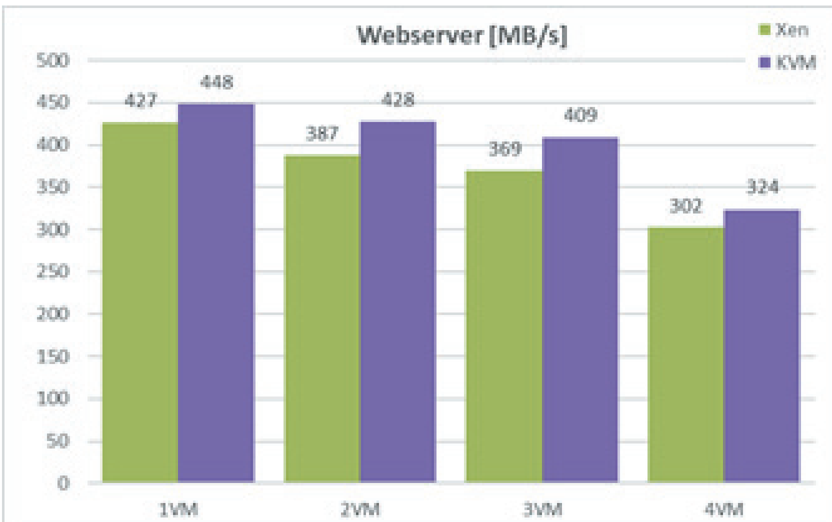

Figure 3. Results from the web server tests

The impact of FS caches on both the guest and host OS for random reads may be minimal, unless the reads are repetitive. For the web server workload, KVM slightly outperforms Xen, with a performance improvement of 5-10%. Observing the achieved web server throughputs and the peak disk speeds reached, both KVM and Xen exhibit relatively high random-read speeds, indicating a solid impact of FS caches on random reads. However, much of the random I/O traffic is still directed to virtual/physical drivers. Towards this type of random read workload, all elements discussed in Chapter 4 are significant. However, the primary components are ***Hyp-proc, VH-proc*** with FS-pair, and the significant cache effects from the FS-cache-pair. Given the numerous cache misses in the two FS caches, the physical disk drivers (host OS) and virtual drivers (guest OS) play a critical role. In the context of the web server workload, we presume that KVM and Xen exhibit similar combinations involving VH-proc with FS cache effects, ***Hyp-proc***, and physical/virtual disk drivers, though KVM shows a slight advantage.

Figure 4. Results from the fileserver tests

Figure 4 illustrates the outcomes of the RFA workload test.

Primarily, the RFA workload is defined by random reads and asynchronous random writes, with a fair volume of input/output requests and a moderate data throughput. The presence of asynchronous writes enhances the importance of FS caches in this workload. In the case of the RFA workload, KVM slightly outperforms Xen, with a performance advantage of 0.1-3%. Observing the achieved RFA throughputs and the peak disk speeds, both of the hypervisors outperform the disk speeds notably. The high RFA throughputs suggest that the guest/host FS caches are the primary influencing factor. In the case of this random read/write workload, all the components discussed in Chapter 4 play an important role. However, we consider that the ***VH-proc*** component (with FS-pair) plays a dominant role, with exceptionally strong cache effects. In the RFA environment, both KVM and Xen demonstrate an excellent combination of ***VH-proc*** with strong FS cache effects, although Xen shows a slight edge.

CONCLUSION

We have evaluated the similarities and differences between the two Linux-based hypervisors, Xen and KVM. According to our mathematical model, the differences between these hypervisors stem from several key factors. While they may initially seem quite similar, a closer analysis reveals substantial differences in hypervisors like Xen and KVM. Despite sharing fundamental components such as ***VH-proc***, ***Hyp-proc***, and ***hFS***, their performance can differ significantly. In our study, the KVM hypervisor excels in fileserver, RFA, and web server workloads, while Xen performs better in mail server workload. These differences are substantial for fileserver workload (2.37-3.92 times), significant for mail server workload (16-62%), relatively minor for web server workload (5-10%), and slight for the RFA workload (0.1-3%).

To reach strong conclusions, it is crucial to perform multiple experiments across different case studies. We propose several potential avenues for future research, including the FS comparison of different type-1 Linux-based hypervisors under various HW configurations and workloads. This includes the analysis of upcoming hypervisor releases to understand any improvements or changes in their capabilities. Additionally, future research could explore various guest OS (including different versions of both Linux and Windows), investigate various FS such as ext4, XFS, and Btrfs, compare alternative benchmarking programs (HD Tune Pro, AS SSD, Fio), and conduct experiments to evaluate the influence of factors like RAM size and the number of CPU cores on performance.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] E. Correia, *Hypervisor based server virtualization*, Encyclopedia of Information Science and Technology, Third Edition, IGI Global, 2015.

[2] A. Kumar, S. Shiwani, "Guest operating system-based performance comparison of VMware & Xen hypervisor," ***Int. J. of Sci, Eng. and Tech***, vol. 2, no. 5, pp. 286-297, 2014.

[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] 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, 2016.

[5] S. Pawar, S. Singh, "Performance comparison of VMware and Xen hypervisor on guest OS," *IJICSE*, vol. 2, no. 3, pp. 56-60, 2015.

[6] C. D. Graziano, *A performance analysis of Xen and KVM hypervisors for hosting the Xen worlds project*, Iowa: Iowa State University, 2011.

[7] V. P. Singh, *Analysis of system performance using VMware ESXi server virtual machines,* PhD Thesis, 2012.

[8] S. Ally, "Comparative analysis of Proxmox VE and XenServer as type 1 open source based hypervisors," *Int. J. of Sci. & Tech. Research,* vol. 7, pp. 72-77, 2018.

[9] 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, 2018.

[10] 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, 2016.

[11] B. Đorđević, M. Piljić, N. Kraljević, V. Timčenko, "Comparison of file system performance in full virtualization with MS Hyper-V and KVM hypervisors," in *Proceedings X ICETRAN*, Novi Pazar, 2022.

[12] B. Đorđević, V. Timčenko, N. Kraljević, N. Maček: *File System Performance Comparison in Full Hardware Virtualization with ESXi, KVM, Hyper-V and Xen Hypervisors*, Advances in Electrical and Computer Engineering, 2021.

[13] B. Djordjevic, V. Timčenko, N. Kraljevic, N. Jovicic, "Performance comparison of KVM and Proxmox Type-1 Hypervisors," in *30th TELFOR*, Belgrade, 2022.

[14] B. Đorđević, D. Sakić, N. Kraljević, V. Timčenko, N. Davidović, "File system performance for type-1 hypervisors on the Xen and VMware ESXi," in *21st INFOTEH-JAHORINA*, East Sarajevo, 2022.

[15] XenServer - https://docs.xenserver.com/en-us/citrix-hypervisor. [Accessed 2023].

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

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