Mathematical Modelling and Case Study with File System Performance Comparison for Linux-based Hypervisors

Borislav Đorđević[1], Kristina Janjić[2], Nenad Kraljević[2]

[1] Mihailo Pupin Institute, University of Belgrade, Volgina 15, 11060 Belgrade, Serbia; e-mail: borislav.djordjevic@pupin.rs

[2] Department of Computer Technologies, The School of Electrical and Computer Engineering of Applied Studies, Vojvode Stepe 283, 11000 Belgrade, Serbia
e-mails: kristinarin5222@gs.viser.edu.rs, nenadk@gs.viser.edu.rs

Abstract: This paper concentrates on comparing the file system performance across various type-1 Linux-based hypervisors. The main contribution of this paper is a mathematical model of the hypervisor-based virtual environment, followed by a real experiment presented as a specific case study. We leverage the model for interpreting the results obtained from the experiment. VMware ESXi, Xen, KVM on Debian-12, and Proxmox VE were selected as the designated hypervisors. Our selection for the benchmark program in testing is Filebench. Recognised for its exceptional flexibility, Filebench enables the simulation of authentic application and server behaviours. CentOS 9, representing the Linux family, fulfilled the role of the guest operating system. The tests in our experiment involved the simultaneous operation of one, two, three, and four virtual machines.

Keywords: hypervisor; ESXi; Xen; KVM; Proxmox; CentOS 9; virtual machine; Filebench

1 Introduction

In the domain of global IT progress, virtualisation emerges as a vanguard of innovation, advancing the management, storage, and utilisation of data and assets. Virtualisation, a technology allowing the simultaneous operation of multiple operating systems on the same hardware platform, elevates system uptime and dependability while optimising resource utilisation. It enables the creation of virtual replicas of computers, servers, and assets, promoting resource optimisation, financial savings, simplified system administration, and enhanced scalability. Virtualisation guarantees the allocation of virtual machines with well-defined CPU, memory, and storage allocations, thus ensuring optimal hardware resource utilisation [1]. Still, virtualisation also solves problems like management complexity, potential expenses related to licensing, security vulnerabilities, and the risk of a single point of failure if a hypervisor or physical server malfunctions. In spite of these obstacles, the advantages of virtualisation typically surpass its disadvantages, solidifying its status as a fundamental component of modern IT infrastructure.

Virtualisation encompasses various classifications, including software, hardware, desktop, data, network architecture, memory, and storage kinds of virtualisation. This paper is focused on hardware virtualisation. A distinguishing feature of hardware virtualisation (also known as server virtualisation or platform virtualisation) lies in its ability to enable virtual machines to operate as autonomous entities, notwithstanding their utilisation of the same underlying physical hardware. Hardware virtualisation encompasses three types: full hardware virtualisation (FHV), para-virtualisation (PV), and partial virtualisation. In full hardware virtualisation, the guest operating system is completely isolated from the virtualisation layer and hardware. Full hardware virtualisation, the focus of this study, involves emulating the entire hardware infrastructure, allowing guest operating systems to be installed and run without any changes. This solution is the most elegant and easiest to use, but the obtainable performances can be low. This issue can be solved using special features of the processor (Intel VT or AMD-V).

In the context of hardware virtualisation, a crucial component is the Virtual Machine Monitor (VMM) or hypervisor software layer. Hypervisors serve to abstract hardware from the operating system, facilitating the concurrent operation of multiple operating systems on the same hardware. Typically, hypervisors are categorised into two classes: type-1 hypervisors, which operate directly on the hardware (known as bare-metal or native hypervisors), and type-2 hypervisors, which operate within the operating system (referred to as hosted hypervisors).

This paper employs four type-1 (bare-metal) hypervisors: VMware ESXi, Xen, KVM on Debian-12, and Proxmox VE. For our choice, we wanted to include all known classes of type-1 Linux-based hypervisors with their modern representatives from each class.

2 Related Work, Objective, and Motivation

In the literature dedicated to virtualisation, a multitude of papers utilises various methods to evaluate the performance of different virtual environments. One common method involves conducting performance comparisons among various hypervisors, including ESXi, Xen, Proxmox, KVM, and MS Hyper-V. Most references include two or three different hypervisors, mostly type-1 or type-2, without mixing of types. Our paper is among the first to include all dominant

Linux-based hypervisors. These papers often feature high-quality experimental case studies and use a range of benchmarks, including AS SSD, HD Tune Pro, Filebench, and Bonnie++, among others [2-14]. The majority of these case studies do not integrate mathematical modelling into their performance evaluation.

Acknowledging that numerous benchmarks in this domain tend to be synthetic, we advocate for Filebench due to its advantageous attributes. Setting itself apart from other contenders, Filebench shows a modern approach, thread-based design, flexibility, adaptability, and ability to simulate the behaviour of real servers and services through the rich Workload Model Language (WML) [15].

The primary focus of this paper is on developing a comprehensive mathematical model specifically aimed at analysing file system performance within a type-1 hypervisor-based virtual environment. Our model encompasses a broad range of input parameters and is open to further refinement. Similar mathematical models are used in [16-19]. These models have an emphasis on hypervisor type-1 virtualisation and include 5 input parameters. The model in this paper is more complex with 7 input parameters and has an accent on hypervisor type-1 virtualisation, but in the context of the Linux architecture on which all used hypervisors are based, which makes this model quite different from the models from [16-19]. Reference [16] covers 4 different hypervisors, references [17] and [18] cover 2 different hypervisors, while reference [19] deals with different versions of the same type-2 hypervisor. The experimental results from [16-18] are quite similar to our paper; there are small deviations in the experimental results, which is a consequence of different hardware and software versions of the hypervisors and host/guest operating systems.

Such an approach, along with the methodology, sets our experiment apart. We adopt a distinctive methodology that begins with formulating a mathematical model, followed by conducting a real experiment serving as a specific case study. Subsequently, the model serves as a main tool in interpreting the results obtained from the experiment, providing a distinct perspective in the assessment of virtual environment performance. On the other side, the obtained experimental results enable the validation of our mathematical model. We concentrated on four dominant Linux-based hypervisors, all of which are globally similar in architecture (Linux), and most use QEMU-based full hardware virtualisation similar in architecture, too. However, despite the similar Linux architecture, analysing the details for all four hypervisors, our model predicts solid differences in many components that will cause differences in FS performance, and the experiment confirms it.

Comparing our paper and similar reference [16], both papers deal with 4 hypervisors; reference [16] includes mixed representatives of Linux architecture (ESXi, Xen, and KVM) and MS Windows architecture (Hyper-V), while this paper includes all possible representatives of Linux-based architectures (ESXi, Xen, KVM, Proxmox). The model in reference [16] universally describes hypervisor-based type-1 virtualisation with 5 input parameters. Our model is more complex with 7 input parameters; it includes hypervisor-based virtualisation, but emphasises the similarities and differences between hypervisors in the context of Linux architecture, which is the main novelty in modelling. Although hardware configurations and software versions are quite different, our paper shows very similar experimental results with reference [16], confirming the superiority of the KVM architecture for complex fileserver workloads, while the same architecture shows quite poor performance in mailserver workloads.

Our experiment involved the utilisation of four bare-metal hypervisors: VMware ESXi, Xen, KVM on Debian-12, and Proxmox VE. These hypervisors were selected for their full hardware virtualisation capabilities, supported by features such as Intel-VT and AMD-V. Each hypervisor was subjected to testing under identical and equitable conditions. CentOS 9, with the XFS file system, was employed as the guest operating system due to its extensive adoption within the Linux family. To conduct the experiments, we employed the Filebench benchmark program, which includes four distinct workloads (fileserver, webserver, mailserver, and random-file-access). After conducting the tests, we validated the results by applying a mathematical model to interpret and authenticate them.

3 Linux-based (type-1) Hypervisors

In the world of Linux-based hypervisor virtualisation, four names stand out prominently: VMware ESXi, KVM (as global architecture), Proxmox, and Xen. Through this narrative, we will explore the characteristics, advantages, and capabilities of each, and we will emphasise that all Linux-based hypervisors are quite similar globally if only full hardware virtualisation is examined.

ESXi (Elastic Sky X integrated) is a type-1 hypervisor (VMware ESXi 8.0 in our paper), directly installed on hardware rather than on top of an operating system. Its architecture comprises the underlying operating system, VMkernel, with processes running above it. ESXi emphasises the following features: type-1 hypervisor and full hardware virtualisation, exclusively utilising VMware's solution for virtual drivers. It employs VMkernel as a hypervisor, and the host operating system represents its own Linux distribution. ESXi also utilises an original VMFS as a host file system, distinct from classical Linux file systems, as it is a cluster-based file system [20].

Xen, another hypervisor, falls under the category of type-1 hypervisors as well (Xen Citrix Hypervisor 8.2.1 in our paper). Xen emphasises the following features: type-1 hypervisor and virtualisation types, including para-virtualisation and full hardware virtualisation. It utilises an open-source QEMU-based solution for full hardware virtualisation, employs the Xen kernel as a hypervisor, and uses the host operating system to represent its own para-virtualized Linux distribution (PV) as Domain0, with support for classical Linux file systems such as ext4 and XFS.

KVM (Kernel-based Virtual Machine) is an open-source virtualisation technology that facilitates hardware-level virtualisation directly within Linux, operating as an integral part of its kernel. Initially initiated through a Red Hat-sponsored effort, KVM has seamlessly integrated into the Linux kernel starting from version 2.6.20, serving as a crucial kernel module [22]. KVM emphasises the following features: type-1 hypervisor and full hardware virtualisation, exclusively using an open-source QEMU-based solution for virtual drivers. It utilises the Linux kernel with a KVM module as a hypervisor and allows host operating systems to represent any Linux distribution. Additionally, KVM employs classical Linux file systems such as ext4 and XFS.

Definitely, KVM is a global architecture, applicable to all modern Linux distributions (installation is made by using a few administrative actions/commands). So, there are a large number of KVM practical implementations, and the different performance is expected on each of them. We used KVM on the Debian Linux distribution: Debian 12 Bookworm, kernel 6.5.11-4, and QEMU emulator version 8.1.2.

Proxmox is one of the KVM implementations; which is practically Debian Linux distribution with a pre-installed KVM option. KVM and Proxmox are of identical architecture; KVM can be employed on any Linux distribution, whereas Proxmox only uses the Debian Linux distributions. Proxmox Virtual Environment (Proxmox VE or PVE) is an open-source server virtualisation platform constructed on the dependable and resilient Debian operating system (Proxmox VE 7.4-1 in our paper). Serving as a bare-metal, type-1 hypervisor, Proxmox VE is installed directly onto physical machines or servers. It presents two distinct virtualisation methodologies: one is hypervisor-driven virtualisation using KVM, while the other is container-driven virtualisation relying on containers (LXC, Linux Containers) [23].

Proxmox emphasises the following features: type-1 hypervisor and full hardware virtualisation, exclusively using an open-source QEMU-based solution for virtual drivers. It uses the Linux kernel with a KVM module as a hypervisor and allows host operating systems to represent some distributions from the Debian Linux family. Additionally, Proxmox employs classical Linux file systems such as ext4 and XFS.

4 Mathematical Model and Hypothesis

In the context of file system performance and access types, workloads generate the typical random and sequential (read/write) components. In the context of file system performance and file system inside, workload features determine the total workload time required to accomplish all operations within the specified file system. These operations include tasks related to file data blocks, directory blocks, various metadata operations, free-list management, housekeeping, and more.

In a virtual environment based on hypervisor technology, there are at least seven significant components that remarkably affect workload time, denoted as Tw_{hyp} in Eq. 1:

$$Tw_{hyp} = f(B, g_k, gFS, VH_{proc}, Hyp_{proc}, h_k, hFS) \tag{1}$$

Surely, in a complex hypervisor-based virtual environment (VE), there are two full operating systems with their own kernels (g_k as the kernel of the guest operating system and h_k as the kernel of the host operating system) and associated file systems (***gFS*** as the guest file system and ***hFS*** as the host file system). They work as a file system pair. Virtual hardware (VH_{proc}) is offered to the guest operating system and hypervisor (Hyp_{proc}) as the kernel for VE, which connects the guest operating system (***gOS***) and host operating system (***hOS***).

The first component, denoted as ***B***, signifies the processing time for benchmarks. ***B*** exhibits consistent attributes across all hypervisors used in our experiments, stemming from the uniform usage of identical benchmarks and their parameters.

The second component, g_k, and the third component, ***gFS***, belong to the guest operating system side. The second component, g_k, denotes the kernel processing time within the guest operating system. g_k exhibits comparable features across all hypervisors used in our experiments. This similarity comes from the consistent utilisation of identical guest virtual machines with identical kernels.

The third component, ***gFS***, denotes the processing time within the guest file system, whereas the guest file system is tightly coupled with the kernel, internal hard disk drivers, and file system caching of the guest operating systems. The ***gFS*** component shows similar characteristics for each hypervisor involved in our experiments. This similarity arises from the consistent utilisation of identical virtual machines and the same guest file systems (XFS in our case).

When we consider guest operating systems as integral components of the virtual environment, the following observations can be made: We use the absolute same virtual machines with all identical parameters (full guest operating system with guest kernel, guest file system, hard disk layout, internal hard disk drivers, and OS/graphical environment). Therefore, a similar impact on performance is expected from the guest side. Differences in the context of the guest side can be influenced by the interaction of the guest operating system with the different virtual hardware (offered by the hypervisor) and the different behaviour of the file system pair due to the differences on the host file system side. In addition, the differences can be influenced by the virtualisation itself, which allows the virtualisation to modify the guest operating systems, from mild patching to serious changes similar to para-virtualisation effects.

The fourth component, ***VH-proc***, represents the processing time allocated to virtual hardware, particularly virtual hard disk drivers. This fourth component, ***VH-proc***, displays notable variations across all hypervisors examined in our experiment. With the exception of Xen, which exclusively employs full hardware virtualisation and favours para-virtualisation (not included in our experiment), all other hypervisors force full hardware virtualisation only. All hypervisors in our experiment, except ESXi, rely on QEMU-based open-source solutions for full hardware virtualisation. Notably, these QEMU solutions offer a wide range of virtual driver sets, spanning from qemu-0.10.0 (released on March 4, 2009) to qemu-8.2.1 (released on January 29, 2024). Consequently, hypervisors such as Xen, KVM, and Proxmox use distinct sets of QEMU virtual drivers, leading to significant variations in the ***VH-proc*** component. ESXi stands apart from the rest by developing its own sets of virtual drivers, renowned for their high quality. As a result, the performance of each hypervisor is anticipated to exhibit considerable diversity. Moreover, ***VH-proc*** is intricately linked with file system caching on both the guest and host operating system sides.

The fifth component, ***Hyp-proc***, signifies the time designated for hypervisor processing. This involves the duration needed for the hypervisor to handle requests from the virtual drivers and subsequently relay these requests to the host operating system. Essentially, file system requests originating from the guest file system are transmitted to the host file system within the VM image file. Across our experiment, we examine four distinct hypervisors: ESXi employs its proprietary VMkernel, XenServer utilises the original Xen hypervisor, while KVM and Proxmox hypervisors are characterised by their unique approach, leveraging real Linux kernels with additional KVM kernel modules. Despite their shared attributes of modernity, slimness, and utilisation of micro-kernel architecture, variations in performance are expected among these hypervisors. Even between KVM and Proxmox, which share identical architectures, differences arise due to the implementation of distinct versions of the Linux kernel and KVM kernel modules. Importantly, all evaluated hypervisors will demonstrate distinct hypervisor processing times.

The sixth component, $\boldsymbol{h_k}$, and the seventh component, ***hFS***, belong to the host operating system side. The sixth component, $\boldsymbol{h_k}$, denotes the kernel processing time within the host operating system. $\boldsymbol{h_k}$ may show very different characteristics for each hypervisor involved in our experiments because all host operating systems are Linux-based and represent different Linux distributions with different versions of the kernel, which will certainly make solid performance differences.

The seventh component, ***hFS***, denotes the processing time of the host file system, which is intricately linked with the kernel, internal hard disk drivers, and file system caching of the host operating system. Notably, we anticipate significant disparities among hypervisors in this aspect. ESXi stands out from other hypervisors by employing its proprietary VMware solution, such as VMFS, a cluster-based file system designed for storing VM image files. Conversely, Linux-based hypervisors typically utilise ext4 or XFS, either with or without the LVM option. Although our evaluation predominantly employed ext4, variations in ext4 versions introduce distinctions. With the ESXi hypervisor and its completely different file system, like VMFS, and with the remaining hypervisors using the popular Linux file systems (ext4 or XFS), but with different versions, the performance will be quite different.

When examining the host operating system within a virtual environment, it becomes evident that all four hypervisors are Linux-based, sharing the same architecture but differing significantly in detail. Each hypervisor adopts its own Linux distribution: ESXi uses a VMware-specific distribution; XenServer employs a Xen-orientated distribution; Proxmox is based solely on the Debian Linux distribution; and KVM is adaptable to various Linux distributions. As a result, variations emerge in host OS kernels, physical drivers, file system versions, and operating system environments. Despite their Linux-based foundation, the hypervisors exhibit substantial disparities in host operating systems and filesystem configurations.

In the context of virtualisation, we highlight the pronounced impact of the fourth and fifth components in Eq. 1, ***VH-proc*** and ***Hyp-proc***. The fourth component, ***VH-proc***, illuminates the efficacy of file system caching on both the guest and host sides, revealing solid differences across all hypervisors.

In summary, across all hypervisors, we can observe both similarities and differences in the following aspects:

On the guest side everything remains consistent, with identical virtual machine features and the benchmark.

On the virtual environment side, while full hardware virtualisation and QEMU solutions for virtual drivers are common (except ESXi), significant differences emerge. Notably, disparities arise in the fourth and fifth components of Eq. 1. ***VH-proc*** and ***Hyp-proc***.

Regarding the host OS side, despite all hypervisors being Linux-based, significant variations are apparent. These differences include kernel versions, Linux distributions, host file systems (especially notable with ESXi use of VMFS and other hypervisors use of ext4 but different versions of ext4), physical hard disk drivers, and file system cache mechanisms.

Within the virtual environment, filesystem pairs are crucial, comprising guest filesystems on host filesystems. In our experiment, we consider filesystem pairs such as XFS on VMFS and XFS on ext4.

With the insights provided by this mathematical model and the identified differences and similarities, we can now interpret the performance results of different hypervisors.

5 HW Test Configuration and Benchmark

To ensure an equitable performance evaluation, it is essential to use identical hardware, operating systems, virtual machines, and measurement methodologies, along with an identical benchmark program. Before initiating testing, it was imperative to establish an appropriate hardware configuration, select the operating system, and determine a benchmark program suitable for all testing iterations. The virtual platforms used included VMware ESXi 8.0, Xen Citrix Hypervisor 8.2.1, Proxmox VE 7.4-1, and KVM on Debian-12 Linux host (Debian 12 Bookworm, kernel host 6.5.11-4, and QEMU emulator version 8.1.2). The entire experiment was carried out on an HP rack server (Table 1). For the guest operating system, CentOS Stream 9 from the Linux distribution family was selected.

Table 1

Test environment (Server)

HPE ProLiant BL460c Gen10	
Component	**Characteristics**
Processor	Intel® Xeon® Silver 4116 CPU @ 2.10GHz,16 MB L3 cache
RAM Memory	32GB DDR4 2400 MHz
Hard disk	2xHPE 480GB SATA3 6G RI SSF SSD as RAID1 Sequential read up to 535 MB/s, Sequential write up to 495 MB/s

All testing procedures employed the benchmark program Filebench 1.4.9.1-3. We use the Filebench, because it is a modern, flexible, multi-threaded filesystem benchmark that perfectly simulates real server environments. This software facilitates the simulation of diverse workloads, enabling a comprehensive performance evaluation of a system. These tests can be customised to suit specific requirements, allowing for the creation of scenarios that mirror real-world situations or experimental cases for analysis. Filebench offers detailed insights into system performance, including file read and write speeds across varying workloads. Two identical hard drives were mounted in a RAID-1 configuration on the HPE ProLiant BL460 Gen10 server. Refer to Table 2 for the parameters of the virtual machines.

Table 2
Virtual machine parameters

Component	Characteristics
vCPU	4
RAM memory	6GB
virtual HDD	/dev/sda with 2 partitions – 64GB
sda1	system partition with CentOS – 32GB
sda2	additional partition for testing – 32GB
Guest operating system	CentOS Stream 9

6 Testing and Results

One of the primaries aims of this study is to evaluate the performance of file systems across four distinct type-1 hypervisors. This evaluation aims to streamline decision making processes for large companies by identifying the most efficient hypervisor model for their needs. Additionally, the study seeks to validate the assumption that an increase in the number of operational virtual machines leads to a noticeable decrease in performance. One component of this research involves employing different types of workloads to obtain performance results. These include fileserver, webserver, mailserver, and random-file-access workloads, all designed to equally evaluate the performance of appropriate servers (Table 3). The duration of each testing session was set to 120 seconds.

Table 3
Parameters of *f files

	Fileserver	***Webserver***	***Varmail***	***RFA***
Nfiles	10000	1000	1000	10000
meandir width	20	20	1000000	20
meanfile size	128k	16k	16k	random
nthtreads	50	100	16	5

Initially, for all four hypervisors (ESXi, Xen, Proxmox, and KVM), the assessment was conducted using a single virtual machine. Subsequently, in the second test, an identical virtual machine was created, and the evaluation was repeated with both virtual machines running simultaneously. This process was then replicated with three virtual machines concurrently running. Finally, the fourth test involved operating four identical virtual machines concurrently.

For all workloads, all components from Chapter 4 are very important. These are ***VH-proc***, guest file system, FS-pair, FS-cache-pair, ***Hyp-proc***, virtual hard disk drivers, physical hard disk drivers, as well as all the mentioned components from the host OS (kernel, host file system, and OS/graphical environments). All these components are different for different hypervisors, especially the VMFS of ESXi as the host file system.

Before the particular results, we will emphasise some facts. For each workload, we firstly measured the native operating system. It is an operating system (CentOS Stream 9 in our paper) in double roles. In the beginning, we installed this operating system on real physical hardware, and we measured the Filebench performance. It is a native or physical operating system, and we use its performance as native or physical performance. Then, we used the same operating system as virtual machines under 4 Linux-based hypervisors, and we measured the Filebench performance as file system performance of virtualisation.

For each workload discussion, we have 4 parts. In the first part, we examine the workload features based on the Workload Model Language (WML) and definition parameters for the given workload in their definition files (Table 3). We estimate the nature of the hard disk dataflow (random/sequential, reading/writing, synchronous/asynchronous) and possible file system cache impact. In the second part, we describe the obtained results without interpretations (which one is the best, etc.). In the third part, we analyse the file system cache impact based on the relation between the workload throughput and maximum possible hard disk speeds (Table 1). And in the fourth part, we interpret the results based on the workload features and mathematical model of Linux-based virtualisation.

6.1 Fileserver Results

The outcomes of the fileserver workload test are presented in Fig. 1, whereas the fileserver speed of the native operating system was 3885.14 MB/s.

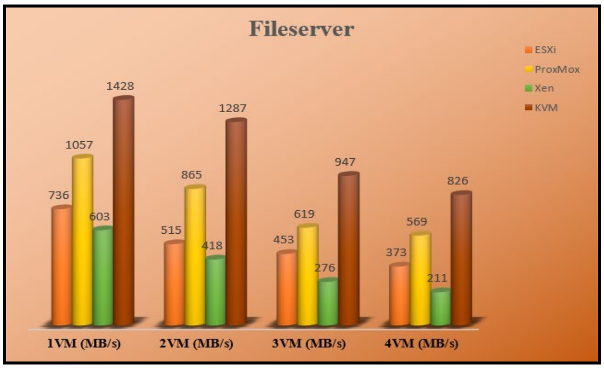

Figure 1
Fileserver test result chart

The characteristics of the fileserver workload encompass the presence of random read, random write, sequential read, and sequential write components. The workload involves a substantial number of I/O operations and a considerable data flow. Given these features, including repetitive reading and diverse types of asynchronous writes, there is a notable impact of file system caches on both the guest operating system and host operating system, particularly in the context of writing operations. In the fileserver workload, the most significant impact is attributed to the fourth component (Eq. 1), ***VH-proc***, working in conjunction with the file system cache on both the guest and host operating systems, primarily due to the large data flow. Additionally, the fifth component (Eq. 1), ***Hyp-proc***, plays a substantial role due to the workload's significant number of I/O operations.

In terms of fileserver workload performance, KVM on Debian-12 stands out as the top choice, followed by Proxmox in second place and ESXi in third, with Xen ranking last. The differences are big; in the duel best/worst as KVM/Xen, KVM on Debian-12 is better than Xen, 2-4 times, the differences are the least on a single virtual machine, while the differences are bigger on a larger number of virtual machines; on four virtual machines, the differences are most pronounced. Compared to the second best, KVM on Debian-12 is better than Proxmox, 35-53%, and the differences are greater on a larger number of virtual machines. Compared to the third-placed ESXi, the best KVM on Debian-12 is better than ESXi by 2-2.5 times, and the rates are higher on a larger number of virtual machines. Certainly, KVM-based solutions are significantly better than the competition. We note that all hypervisors have solid performance drops compared to the native host (best case: Native/1VM), KVM on Debian-12 2.7 times, Proxmox 3.7 times, ESXi 5.3 times, and Xen 6.4 times.

When examining the achieved throughput of the fileserver workload alongside the maximum speeds of the hard disk interface (approximately 600 MB/s) and the maximum sequential speeds of SSD hard disks (around 500 MB/s), the conclusion is evident: with a single virtual machine, all hypervisors exhibit throughput surpassing the maximum hard disk speeds. However, when two virtual machines are running, ESXi and Xen begin to falter, with their throughputs dropping below the maximum hard disk speeds. Proxmox maintains satisfactory performance up to three virtual machines, but with four virtual machines, its throughput falls below the maximum hard disk speeds. KVM on Debian-12 is by far the best; it has better throughputs than the maximum hard disk speeds on all virtual machines. High throughput rates indicate the dominance of file system cache pairs (guest/host caches), alongside a significant volume of I/O requests directed to the virtual-physical drivers, exhibiting both random and sequential features. These high throughputs, surpassing hard disk speeds, underscore the effectiveness of FS cache pairs, particularly evident with KVM/Proxmox setups but to a lesser extent with ESXi and Xen, where virtual/physical hard disk drivers take precedence.

In terms of fileserver workload, we believe that the main components are primarily ***VH-proc*** with FS-pairs and with the cache effects of this FS-cache-pair, but due to a high frequency of cache misses in both the guest and host caches, the significance of hard disk drivers within the guest operating system. Virtual hard disk drivers of the hypervisors and physical hard disk drivers of the host operating system are also important. In the context of a complex workload characterised by substantial data flow, such as the fileserver, we assume that the KVM on Debian-12 and Proxmox generation possesses the best combination of components: ***VH-proc*** with file system cache effects of FS-cache-pair, as well as a combination of guest/virtual/physical hard disk drivers. KVM on Debian-12 is by far the best, closely followed by Proxmox, which has a very similar architecture. On the other hand, ESXi and especially Xen have a weaker mention combination and probably a weaker hit/miss ratio in file system cache-pair.

6.2 Mailserver Results

The outcomes of the mailserver workload test are displayed in Fig. 2, whereas the mailserver speed of the native operating system was 187.20 MB/s.

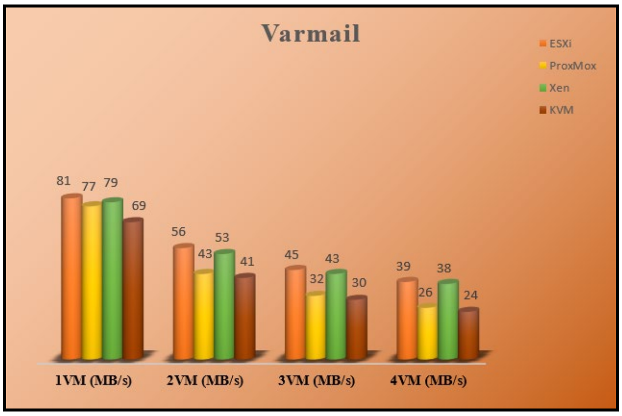

Figure 2
Varmail test result chart

The characteristics of the mailserver workload include a dominance of random read and synchronous random write components. The workload entails a moderate number of I/O operations and a moderate data flow. Given these features, particularly the presence of random reading and synchronous random write components, the influence of file system caches on both the guest operating system and host operating system is minimal. In the context of the mailserver workload, the primary influence stems from the fourth component (Eq. 1), ***VH-proc*** (excluding the impact of file system cache), attributed to the workload's moderate data flow. Additionally, the fifth component (Eq. 1), ***Hyp-proc***, has a substantial effect due to the workload's moderate number of I/O operations.

In terms of mailserver workload performance, ESXi stands out as the top choice, followed by Xen in second place and Proxmox in third, with KVM on Debian-12 ranking last. The differences are not big; in the best/worst duel (ESXi vs. KVM), ESXi is better than KVM on Debian-12 by 18-67%, the differences are the smallest on a single virtual machine, and the differences are bigger on a larger number of virtual machines. Compared to the second best, ESXi is better than Xen by 1-5%, and the differences are small; these two hypervisors are very similar. Compared to the third place, the best ESXi is better than Proxmox by 5-54%, and the scales are higher on more virtual machines. Certainly, KVM-based solutions are significantly weaker than the competition, and ESXi and Xen are very similar. We note that all hypervisors have solid performance drops compared to the native host (best case: Native/1VM), ESXi 2.32 times, Xen 2.36 times, Proxmox 2.43 times, and KVM on Debian-12 2.73 times.

When examining the achieved throughput of the mailserver workload alongside the maximum speeds of the hard disk interface (approximately 600 MB/s) and the maximum sequential speeds of SSD hard disks (around 500 MB/s), the conclusion is evident: on all virtual machines, all hypervisors have significantly lower throughputs than the maximum hard disk speeds. The low mail throughput speeds for email operations suggest minimal impact from guest/host caches within cache pairs, indicating that the majority of I/O requests are directed towards virtual-physical drivers. This implies that the FS cache pairs did not significantly intercept hard disk requests, resulting in nearly all the random read/write operations being handled through virtual/physical drivers.

Nevertheless, we assert that the principal components primarily consist of ***Hyp-proc*** and ***VH-proc*** with FS-pair, although with minimal cache effects from FS-cache-pair. As both caches in the pair exhibit weak influence, the hard disk drivers of the guest OS, the virtual hard disk drivers of the hypervisor, and the physical hard disk drivers of the host OS assume significant importance for random read/random write performance. In the context of a data flow predominantly characterised by random read and synchronous random write, our assumption is that the ESXi and Xen present the most favourable combinations: a combination of ***Hyp-proc*** and ***VH-proc*** with minimal file system cache effects and a combination of guest/virtual/physical hard disk drivers. Furthermore, the ESXi and Xen exhibit similar best combinations, whereas the Proxmox and KVM on Debian-12 demonstrate a somewhat weaker combination for the mailserver workload.

6.3 Webserver Results

The outcomes of the webserver workload test are presented in Fig. 3, whereas the webserver speed of the native operating system was 476.20 MB/s.

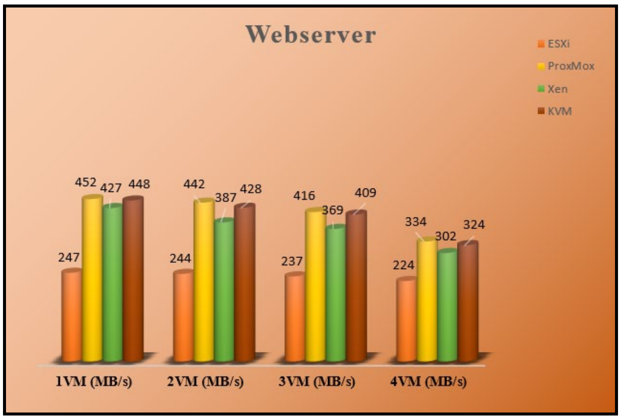

Figure 3
Webserver test result chart

The characteristics of the webserver workload include a predominant presence of random read and small random write components. The workload involves a moderate volume of I/O operations and highlights a limited data flow. Given these attributes, particularly the prevalence of random read components, the influence of file system caches on both the guest operating system and the host operating system is small, except for repeating reading. In the context of the webserver workload, the most significant impact is attributed to the fifth component (Eq. 1), ***Hyp-proc***, due to the workload's moderate number of I/O operations. Additionally, the fourth component (Eq. 1), ***VH-proc*** (with a limited impact of file system cache), plays a substantial role owing to the workload's small data flow.

In terms of webserver workload performance, Proxmox stands out as the top choice, followed by KVM on Debian-12 in second place and Xen in third, with ESXi ranking last. The differences are not big; in the best/worst duel (Proxmox vs. ESXi), Proxmox is better than ESXi by 83-49%, and the differences are the smallest on 4 virtual machines, and the differences are bigger on a smaller number of virtual machines (cache effect). Compared to the second best, Proxmox is better than KVM on Debian-12 by 1-3%, and the differences are small; these two hypervisors are very similar. Compared to the third place, Proxmox is better than Xen by 6-14%, and the differences are bigger on a larger number of VMs. Surely, KVM-based solutions are significantly better than the competition, and Proxmox and KVM on Debian-12 are very similar. We note that all hypervisors have a slight drop compared to the native host (best case: Native/1VM), Proxmox 5.3%, KVM on Debian-12 6.2%, Xen 12%, and only ESXi has solid drops from 93% to 2 times.

When examining the achieved throughput of the webserver workload alongside the maximum speeds of the hard disk interface (approximately 600MB/s) and the maximum sequential speeds of SSD hard disks (around 500MB/s), the conclusion is evident: on all virtual machines, all hypervisors have quite good random-read throughput close to the maximum hard disk speed, except for ESXi, which is significantly weaker. The relatively high webserver throughputs suggest the influence of guest/host caches for random reads, although a significant number of I/O requests with random read characteristics are directed towards virtual-physical drivers. Achieving high throughput for random read workloads close to maximum hard disk speeds indicates some success of the FS cache-pair with KVM/Proxmox/Xen, but not with ESXi. With ESXi, there is a weaker cache effect, and it is more influenced by guest/virtual/physical hard disk drivers. This implies that while the FS cache pair absorbs a certain amount of random read hard disk requests, a considerable number of random read IO operations (as cache misses) were processed through guest/virtual/physical drivers.

However, we believe that the primary components consist of ***Hyp-proc***, ***VH-proc*** with FS-pair, and the limited cache effects of FS-cache-pair. With numerous misses in both caches, the significance of the hard disk driver of the guest OS, the virtual driver of the hypervisor, and the physical hard disk drivers of the host OS, particularly regarding their random read performance, becomes paramount. In a data flow predominantly characterised by random reads, our assumption is that Proxmox/KVM/Xen hypervisors present the most favourable combination of components: ***VH-proc*** with limited reading file system cache effects, ***Hyp-proc***, and guest/virtual/physical hard disk drivers. The ESXi exhibits the least optimal combination of ***VH-proc***, ***Hyp-proc***, and guest/virtual/physical hard disk drivers. ESXi has a very different host file system (VMFS) and therefore a very different FS pair (XFS-on-VMFS), which also affects its very poor webserver performance.

6.4 RFA Results

The results for the RFA workload test are illustrated in Fig. 4, whereas the RFA speed of the native operating system was 16700.44 MB/s.

The key attributes defining the RFA workload include the prevalence of random read and asynchronous random write components. A moderate volume of I/O operations and a balanced data flow characterise the workload. Given these distinctive features, particularly the inclusion of asynchronous random writing, the influence of file system caches is noteworthy on both the guest and host operating systems. The primary influence on RFA workload stems from the fourth component (Eq. 1), ***VH-proc***, characterised by a large impact from file system cache, attributed to a moderate writing dataflow. Additionally, the fifth component (Eq. 1) and ***Hyp-proc*** exhibit a significant impact, driven by a moderate number of I/O operations.

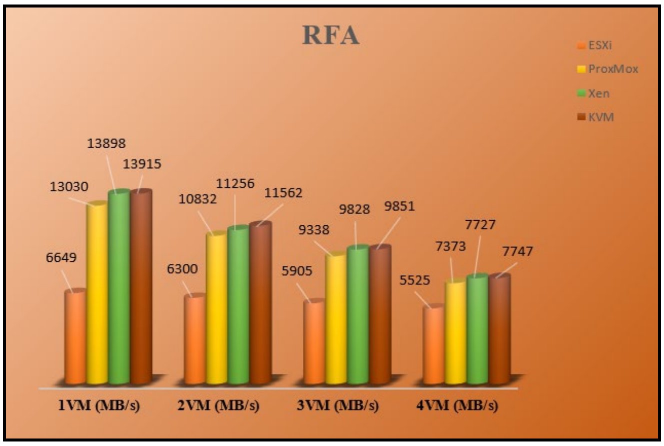

Figure 4
RFA test result chart

In terms of RFA workload performance, KVM on Debian-12 stands out as the top choice, followed by XEN in second place and Proxmox in third, with ESXi ranking last. The differences are big; in the best/worst duel, KVM on Debian-12 is better than ESXi from 40% to two times, and the differences are the least on four virtual machines, and the differences are bigger on a smaller number of virtual machines (cache effect). Compared to the second best, KVM on Debian-12 is better than Xen by 1-3%, and the differences are small; these two hypervisors are very similar. Compared to the third place, the best KVM on Debian-12 is better than Proxmox by 7-5%, and the rates are higher on a smaller number of virtual machines. Surely, KVM-based solutions are quite good, as are Xen-based solutions, KVM/Xen/Proxmox are very similar, and ESXi is much weaker. We notice that all hypervisors have a small attenuation compared to the native host (best case Native/1VM), KVM on Debian-12 20%, Xen 20.1%, Proxmox 28%, and only ESXi has solid drops about 2.5 times.

When examining the achieved throughput of the RFA workload alongside the maximum speeds of the hard disk interface (approximately 600 MB/s) and the maximum sequential speeds of SSD hard disks (around 500 MB/s), the conclusion is evident: on all virtual machines, all hypervisors have significantly higher throughput than the maximum hard disk speeds. High RFA throughputs indicate that guest/host caches exert significant influence, with only a few I/O requests directed towards virtual-physical drivers featuring random read/write characteristics. Achieving throughputs higher than hard disk speeds indicates notable success of the FS cache-pair with KVM/Xen/Proxmox setups, although with slightly less success with ESXi. This implies that the FS caches absorbed almost all random read/random write hard disk requests.

However, we assert that this time the component that absolutely dominates is ***VH-proc*** with file system pair, and with the cache effects of file system cache in pair. In scenarios dominated by random reads and asynchronous random writes data flow, the KVM/XEN/Proxmox displays the most favourable combination with a ***VH-proc*** featuring strong file system cache effects. The ESXi shows the least desirable combination of ***VH-proc with*** FS-pair, and with the cache effects of FS-cache-pair, it probably has the lowest hit/miss ratio, but that is a consequence of the ESXi having a very different host OS FS (VMFS) and therefore a very different FS pair (XFS-on-VMFS), which causes very poor RFA performance.

Conclusion

This paper is focused on comparing the file system performance of specific type-1 Linux-based hypervisors, the four most dominant globally: ESXi, Xen, KVM on Debian-12, and Proxmox. At first glance, hypervisors are very similar in architecture, especially KVM on Debian-12 and Proxmox. We have discerned the similarities and differences among each individual Linux-based hypervisor. Our mathematical model suggests that these distinctions arise from several key components: hypervisor processing, virtual hardware processing, file system caching on both guest and host sides, host filesystems, and components of the host operating system (kernel and OS/graphical environments). While these components may initially seem very similar, upon closer examination, they reveal significant differences among Linux-based hypervisors.

In this case analysis, representatives of the KVM on Debian-12 and Proxmox hypervisors excel in the fileserver, RFA, and webserver workloads, but perform poorly in the mailserver workload. The Xen hypervisor's representative performs relatively well in the mailserver and RFA workloads but poorly in the fileserver workload. The ESXi hypervisor's representative performs relatively well in mailserver, poorly in the fileserver workload, and very poorly in webserver and RFA workloads. Clearly, for these four workloads, KVM demonstrates the most favourable performance, while ESXi exhibits the least favourable performance, likely due to VMFS as its host file system, which is totally different from others.

The paper underscores the significance of selecting the right hypervisor to achieve optimal performance for the specific workloads. However, this is only one case study, so we do not guarantee that it will behave exactly like this on other hardware configurations with Linux-based hypervisors. Still, for some options, like complex fileserver workloads, our results agree with other references, where they confirm the superiority of KVM architecture for fileserver performance. For serious conclusions, a large number of different experiments should be provided that represent different case studies and which can be used to create a KDB (Knowledge Data Base) about the behaviour of Linux-based hypervisors. But our math model is universal and applicable to most experimental case studies.

We present potential options for future research, such as exploring the differences between various versions and generations of Linux-based hypervisors within comparable configurations and workloads. This encompasses evaluating forthcoming releases of Linux-based hypervisors, assessing a range of guest operating systems (including various versions of Linux and Windows), scrutinising different file systems such as ext4, Btrfs, and XFS, and comparing against alternative benchmark tools like Fio, HD Tune Pro, and AS SSD. Additional tests that can be conducted on factors that may affect performance are RAM memory allocation and the use of additional CPU cores.

Acknowledgements

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

References

[1] E. Correia, Hypervisor based server virtualisation, 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," IJSET, Engineering and Technology, Vol. 2, No. 5, pp. 286-297, 2014, ISSN: 2348-4098

[3] S. Pawar, S. Singh, "Performance comparison of VMware and Xen hypervisor on guest OS," IJICSE, Vol. 2, No. 3, pp. 56-60, 2015, ISSN: 2393-8528

[4] H. Kazan, L. Perneel, M. Timmermann, "Benchmarking the performance of Microsoft Hyper-V server, VMware ESXi and Xen hypervisors," IJETCIS, Vol. 4, No. 12, pp. 922-933, 2013, ISSN 2079-8407

[5] A. Bhatia, G. Bhattal, "A comparative study of various hypervisors performance," IJSER, Vol. 7, No. 12, pp. 65-71, 2016

[6] V. P. Singh, "Analysis of system performance using VMware ESXi server virtual machines" PhD Thesis, 2012, Online: http://hdl.handle.net/10266/1809

[7] C. D. Graziano, "A performance analysis of Xen and KVM hypervisors for hosting the Xen worlds project," Iowa State University (2011) Graduate Theses and Dissertations. 12215. doi:10.31274/etd-180810-2322

[8] S. A. Algarni, M. R. Ikbal, R. Alroobaea, A. S. Ghiduk, F. Nadeem, "Performance evaluation of Xen, KVM, and Proxmox hypervisors", IJOSSP, Vol. 9, No. 2, 2018, doi:10.4018/IJOSSP.2018040103

[9] V. K. Manik, D. Arora, "Performance comparison of commercial VMM: ESXi, XEN, HYPER-V & KVM," 4th Int. Conf. on Computing for Sustainable Global Development, 2016, ISBN:978-1-4673-9417-8

[10] S. Ally, "Comparative analysis of Proxmox VE and XenServer as type 1 open source based hypervisors", IJSTR©2018, Volume, 7(3), 72-77, 2018, ISSN 2277-8616, http://www.ijstr.org/final-print/mar2018/Comparative-

Analysis-Of-Proxmox-Ve-And-Xenserver-As-Type-1-Open-Source-Based-Hypervisors-.pdf

[11] A. Kovari, P. Dukan, “KVM & OpenVZ virtualisation based IaaS open source cloud virtualisation platforms: OpenNode, Proxmox VE”, In 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics (pp. 335-339) IEEE, Subotica, September 2012, doi: 10.1109/SISY.2012.6339540

[12] S. Lozano, T. Lugo, J. Carretero, “A Comprehensive Survey on the Use of Hypervisors in Safety-Critical Systems” IEEE Access, Open Access, Vol. 11, pp. 36244-36263, 2023, ISSN: 21693536, doi: 10.1109/ACCESS.2023.3264825

[13] E. Gamess, M. Parajuli, S. Shah, “PERFORMANCE EVALUATION OF THE KVM HYPERVISOR RUNNING ON ARM-BASED SINGLE-BOARD COMPUTERS”, International Journal of Computer Networks and Communications, Open Access, Vol. 15, Issue 2, pp. 147-164, March 2023, ISSN 09752293, doi: 10.5121/ijcnc.2023.15208

[14] J. Jiménez, L. Muguira, U. Bidarte, A. Largacha, J. Lázaro, “Specific Electronic Platform to Test the Influence of Hypervisors on the Performance of Embedded Systems”, Technologies, Open Access, Vol. 10, Issue 3, June 2022, Article number 65, ISSN 22277080, doi: 10.3390/technologies10030065

[15] Filebench, https://github.com/filebench/filebench [Accessed: 2024]

[16] B. Đorđević, V. Timčenko, N. Kraljević, N. Maček: “File System Performance Comparison in Full Hardware Virtualisation with ESXi, KVM, Hyper-V and Xen Hypervisors”, Advances in Electrical and Computer Engineering, Vol. 21, Iss. 1, 2021 doi: 10.4316/AECE.2021.01002

[17] B. Djordjevic, V. Timčenko, N. Kraljevic, N. Jovicic, "Performance comparison of KVM and Proxmox Type-1 Hypervisors“, 30th TELFOR 2022, Nov 15-16, 2022, Belgrade, Serbia, pp. 441-440, IEEE, doi: 10.1109/TELFOR56187.2022.9983666

[18] B. Đorđević, M. Piljić, N. Kraljević, V. Timčenko, “Comparison of file system performance in full virtualisation with MS Hyper-V and KVM hypervisors”, pp. 565-568, X ICETRAN, ISBN 978-86-7466-930-3, June 6-9, 2022, Novi Pazar, Serbia

[19] B. Đorđević, S. Gucunja, N. Kraljević and N. Davidović, "Performance comparison of different hypervisor versions of the type-2 hypervisor VirtualBox“, X ICETRAN, East Sarajevo, B&H, June 5-8, 2023, doi: 10.1109/IcETRAN59631.2023.10192192

[20] ESXi VMware, https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/ESXi_architecture.pdf [Accessed: 2024]

[21] Xen, https://docs.xenserver.com/en-us/citrix-hypervisor [Accessed: 2023]

[22] KVM, https://ubuntu.com/blog/kvm-hyphervisor [Accessed: 2023]

[23] W. Ahmed, "Mastering Proxmox: Build virtualized environments using the Proxmox VE hypervisor", Packt Publishing Ltd, 2017