Virtualization Maturity in Creating System VM: An Updated Performance Evaluation

Daniel Silva, João Rafael, Alexandre Fonte*
School of Technology, Polytechnic Institute of Castelo Branco, Portugal
Email: dsilva@ipcbcampus.pt, j.rafael@ipcbcampus.pt, adf@ipcb.pt
* Corresponding author

Abstract: **Virtualization technologies are indispensable in operating data centers and supporting cloud infrastructures, providing cost reduction (CapEx and OpEx), high availability, and disaster recovery. Hypervisor-assisted virtualization is one of the leading virtualization technologies, with the hypervisor being the software layer responsible for presenting the virtualized view of the hardware to system-level VMs. However, the virtualization overhead it introduces has implications into the computing infrastructure performance.**

This paper revisits key concepts about virtualization, technologies and techniques, types of VMs and hypervisors, and provides an up-to-date comparison between native and VM environments using workload metrics such as CPU and memory scores, disk speed, and network throughput to determine virtualization overhead. Our results show a clear overall trend toward meritorious performance and the maturity of the technologies used to create system-level VMs.

Keywords: **virtualization, virtualization technologies, virtual machine, hypervisor.**

I. INTRODUCTION

The growing demand for highly available computing resources (HA) and the widespread adoption of Cloud computing have driven the adoption of virtualization technologies. The benefits offered by virtualization in controlling total acquisition costs (CapEx), and lower infrastructure maintenance costs (OpEx) have contributed to this interest, and it has become one of the core building blocks of large data centers and cloud infrastructure [1,2].

In the 1970s, IBM pioneered a set of robust time-sharing solutions for its mainframes, logically partitioning them into what we now call Virtual Machines (VM). In the 1990s, with the rise of x86 and the Windows operating system (OS) and the emergence of Linux as a server OS, virtualization was boosted, now as a broader solution adopted by end users or organizations for running multiple OS or addressing low infrastructure utilization. In this period VMware was the key contributor to the advances in virtualization by introducing it in x86, overcoming the challenges raised by this ISA [3].

Today, large data centers rely on virtualization techniques to abstract physical hardware, creating large pools of logical resources such as CPU, storage, and networks. In these environments, virtualization offers effective solutions, such as Live Migration for VM consolidation with zero downtime, Virtual Failover Clustering for high availability, or cloning VMs for Disaster Recovery and ensuring Business Continuity [4].

In the cloud, virtualization is important to ensure that the resources needed for Cloud services are aggregated and delivered to end users quickly and automatically [5,6]. In a typical Cloud scenario, the VM might be subscribed and provided on demand, as an Infrastructure Service (IaaS). Also, in the Platform as a Service (PaaS) model, a full development and execution environment can be subscribed by developers.

Modern hypervisors play a critical role in the operation of large data centers and the Cloud [7,8]. Hypervisors are responsible for presenting the virtualized view of the hardware to VMs. By efficiently assigning the required resources to VMs, such as CPU, memory, and storage, hypervisors enable one or more operating system (OS) instances to share the hardware of a single host system.

There are two main types of hypervisors: type I and type II. A type I or bare metal hypervisor is designed to operate directly on the hardware/firmware layer, while a type II hypervisor requires the presence of the host operating system and executes on it, similar to other OS processes. However, it's important to note that the virtualization overhead added by hypervisors has implications into the performance of the computing infrastructure.

This paper aims to evaluate this form of hypervisor-assisted virtualization and compare the performance of modern hypervisors that do not require OS modification, specifically native (type I) and host-based (type II) hypervisors. We will analyze and discuss the result of each benchmark using workload metrics, such as CPU and memory scores, disk read/write speeds, network packet losses and throughput. The objective is to assess the level of overhead of virtualization and its impact on performance.
The main contributions of this work are:

(1) Reviewing key virtualization concepts, technologies, and techniques, the VMs and hypervisors, and their types, resource virtualization, and security issues.

(2) Providing an up-to-date comparison between native and system-level VMs using recent hardware and software, benchmarks, and proper workloads.

(3) Demonstrating the maturity of hypervisor-based virtualization used in creating and managing system VMs.

While this paper primarily focuses on the above contributions, it acknowledges that there are other important aspects in the field of virtualization, such as security, and scalability, which could be further explored. However, due to scope limitations, the paper does not delve extensively

into these specific challenges and issues.

The rest of this paper is organized in 4 sections. Section II presents the necessary background. Then, Section III presents the scope of this work, and related word. Sections IV and V describe the experimental testbed, the virtualization and benchmark tools adopted; and present the evaluation results. Finally, Section VI presents the conclusions.

II. BACKGROUND

This section revisits the virtualization concepts, HW virtualization technologies, types of VMs (Virtual Machine) and hypervisors, resource virtualization, and security issues.

A. *Concept of Virtualization, VM and Hypervisor.*

The term of virtualization is widely described as a technique that allows the creation of a representation of one or more isolated machines as if they were real, which can be run on a single host computer. This concept of VM was formalized in a pioneering paper in [8], as "*an efficient, isolated duplicate of the real machine*". The present work specifically focuses on system-level VMs, or System VMs.

The virtualization software layer that sits between the host system hardware and the VM, is referred to as the virtual machine monitor (VMM) or hypervisor. The term VMM was also introduced in [8], which established three formal performance requirements for a VMM:

(1) Ensure that an application running on a system VM behaves identically to the real environment.

(2) Ensure efficient execution of instructions, a dominant subset directly on the host CPU.

(3) Have control over the host hardware resources.

Virtualization assisted with hypervisors enables the encapsulation and isolation of workloads into VMs, which can then be consolidated into a single host system, resulting in improved hardware utilization [4]. With the advent of multi-core CPUs, it has become even more effective, enabling numerous VMs to be assigned to a single physical node, resulting in higher resource utilization rates and lower power consumption compared to single-core [9].

In summary, virtualization provides IT infrastructures and Cloud, with lower CapEx and OpEx costs, and can achieve four major design goals: scalability, reliability, dependability, and security compliance [1,2,6].

B. *Hardware Virtualization Technologies*

HW virtualization can be classified into two types: Native and Hosted types depending on the technologies employed. Native virtualization includes full virtualization, para-virtualization, and HW-assisted virtualization [10-12].

It is worth remembering the concepts of ring protection on x86, and of ISA (Instruction Set Architecture). The x86 processors have four rings or privilege levels, with ring 0 being the most privileged, and ring 3 the least. The OS manages hardware resources, handles system calls, and executes the privileged instructions in ring 0, while user level instructions are executed in ring 3 [6,13].

In turn, the ISA is part of the abstract model of a computer, defining what the CPU can do and the machine code behavior. It defines the supported data types, the registers, the features and memory managing, the instruction set, and the input/output model [14,15].

***Full-Virtualization*:** In full virtualization, the guest OS runs in a native VM environment that simulates all hardware, without requiring any modifications. The hypervisor (a type I, see section C) is installed on top of the underlying host hardware layer, and runs in Ring 0 (see Fig. 1). The hypervisor is directly responsible for managing the VMs, providing each OS with the virtualized hardware resources.

This type of virtualization provides an isolated execution environment with high performance, primarily due to the use of a Type I hypervisor [13]. Guest OSs only need to access drivers that are usually generic or specialized for optimal performance or access to advanced hardware services.

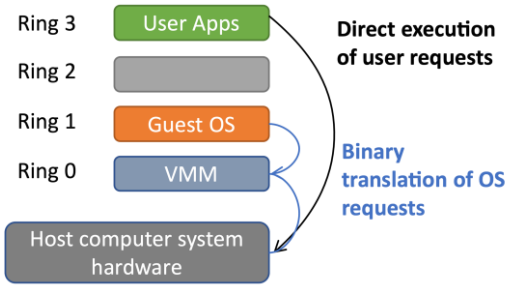

Fig. 1. Full-Virtualization model. source: VMWare [12].

Para-virtualization or OS-Assisted Virtualization: In this form, the hypervisor, also a type I, is also installed on the host system hardware. However, it requires modifications to the guest OS. These modifications are focused on the OS kernel, enabling it to communicate more efficiently with the hypervisor using *hypercalls*. In other words, para-virtualization replaces privileged or sensitive instructions with *hypercalls*, as in Fig. 2. Hypervisor can also provide *hypercalls* for other kernel operations (e.g., memory management and interrupt handling).

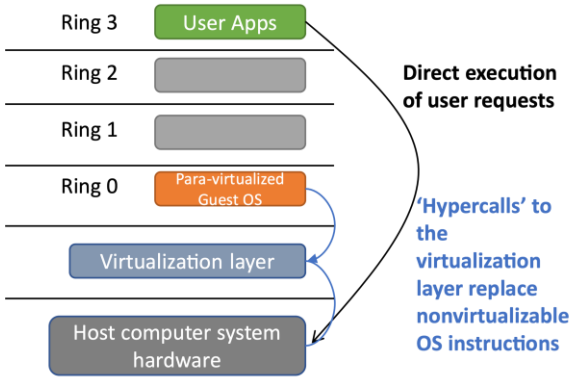

Fig. 2. Para-virtualization model. source VMware [12].

Paravirtualization is very efficient and outperforms full virtualization [13]. However, it also carries limitations, such as limited portability to certain types of HW, or not be able of taking full advantage of HW, such as GPUs. It has also limited OS support by requiring its modification, especially by proprietary OSes, and the implementation requires in-depth knowledge and effort in modifying the OS and creating specialized drivers.

Hardware-assisted Virtualization: HW-assisted virtualization uses the HW potential of the CPU and I/O

devices. This type of virtualization requires no modifications to the OS and allows the instructions of the various VMs to be executed natively on the CPU. A type 1 hypervisor operates in a new root level mode that is lower than the OS kernel level (or level -1). Non-virtualizable instructions are handled via trap-and-emulate by the hardware, as in Fig. 3. Binary translation of instructions ceases to exist, improving VM performance.

Examples of hardware-assisted virtualization technologies are, Intel VT-x and AMD-v, introduced by Intel and AMD for x86 [14-16]. These technologies were specifically designed to mitigate the performance problems experienced in full virtualization and para-virtualization due to the overhead of hypervisors when virtualizing x86 hardware.

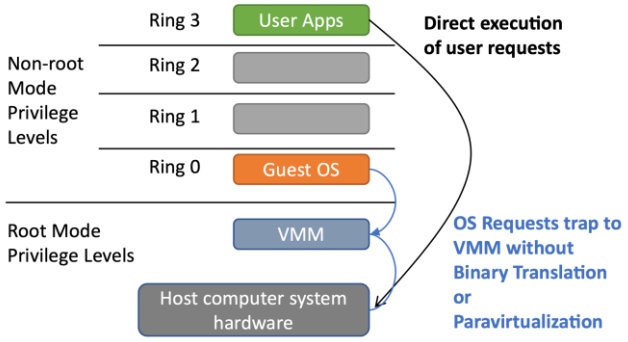

Fig. 3. Hardware-assisted Virtualization model. source VMware [42].

C. *Types of VMs and Hypervisors*

There are essentially two types of system-level VMs: Native VMs and hosted VMs, which are also corresponding to two types of hypervisors [6,7,11]. Next, we discuss the two types of hypervisors, including examples of hypervisors classified by CPU virtualization techniques, and the advantages and disadvantages of each type.

Type 1 hypervisor: A type 1 hypervisor, also called as native or bare-metal hypervisor, runs directly on the host hardware, as illustrated in Fig. 4. The two main advantages of a type 1 hypervisor are to provide:

(1) High performance: They have direct access to the HW and control of its resources, rather than running inside an OS, reducing HW overhead.

(2) Highly secure environment: They are not subject to the flaws and vulnerabilities that are endemic to OSs. Advanced security features such as data encryption, network isolation, and granular access control ensure each VM is isolated from malicious software activity.

There are several examples of type 1 hypervisors, from which three groups can be identified:

- HW-assisted Type I hypervisors: *KVM, Microsoft Hyper-V and VMware ESXi.*
- Type I Hypervisors using the DBT: *VMware ESXi.*
- Type I Hypervisors supporting para-virtualization: *Xen, KVM, VMware VMI.*

Type 2 Hypervisor: A type 2 hypervisor, or hosted hypervisor, is installed and runs as an application on the host OS. When the guest OS attempts to access the virtualized HW resources, the hypervisor acts as an intermediate layer that relies on the host OS to perform the operations of managing CPU calls, memory, and storage management, as illustrated in Fig. 5.

Type 2 hypervisors can be further categorized into three groups, similar to type 1 hypervisors:

- HW-assisted Type II hypervisors: *VMware Workstation/Fusion, Parallels, and Oracle Virtual Box.*
- Type II Hypervisors using the DBT: *VMware Workstation/Fusion, Parallels, Oracle Virtual Box, and Microsoft Virtual PC.*
- Type II Hypervisors that support para-virtualization: *VMware Workstation added with specialized drivers.*

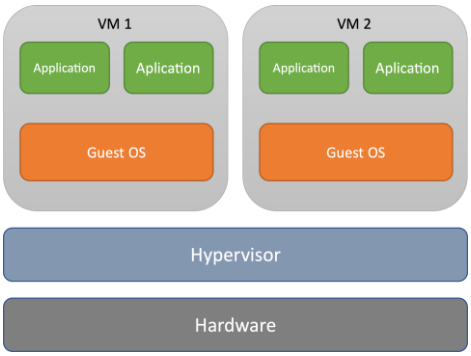

Fig. 4. Hypervisor Type 1 model.

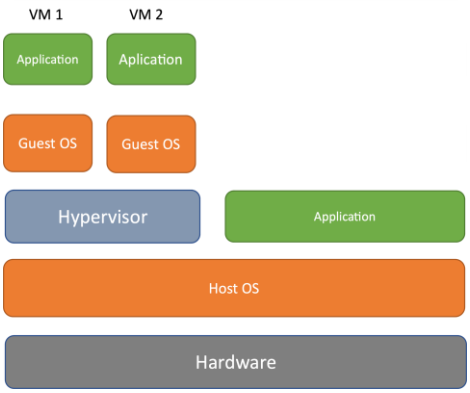

Fig. 5. Hypervisor Type 2 model.

D. *Resource Virtualization*

Next, the main resources that can be virtualized and delivered to system-level VMs will be discussed.

1) *CPU Virtualization*

CPU virtualization presents multiple copies of the CPU (vCPUs), or cores in multi-core systems, to one or more different OSes. Different techniques can be used to accomplish CPU virtualization, as it follows.

Hardware emulation is effective when the presented ISA architecture differs from the host hardware, but it is inefficient as all instructions are interpreted. A hypervisor usually doesn't support different ISA, it requires an emulator (e.g., QEMU) [17]. These Codesigned VMs, are not the focus of the present work.

When the ISA architecture presented is the same, dynamic binary translation (DBT) of instructions is a highly effective technique. The hypervisor inspects blocks of instructions guest OS attempts to execute and translates them into the target instructions. Performance is improved since the code for a statistically dominant set of instructions (the user-level

instructions) is executed directly on the host CPU, while the rest of code with critical instructions is carefully handled and cached by the hypervisor.

In case of CPU virtualization being hardware-assisted (HW), the HW provides virtualization support and simplifies the management of VMs, so that the trap of privileged or sensitive instructions is done automatically for the hypervisor. Modern CPUs thus allow instructions from multiple VMs to execute and share the host CPU.

Intel VT-x and AMD-V are two x86 virtualization extensions that enable HW-assisted CPU virtualization and improve VM performance [14][15]. Their goal was to make the x86 virtualizable, ensuring x86 complies with the virtualization requirements listed in section A, and to simplify the development of hypervisors by removing the complexity of DBT:

Intel VT-x: Intel VT-x (Intel Virtualization Technology) is a set of virtualization extensions for Intel's x86 processors. These allow VMs to direct access to physical system resources, bypassing the virtualization software layer, resulting in improved performance and efficiency. Intel VMX is a set of instructions added later to VT-x, which are intended to allow the virtualized OS to enter and exit a privileged, level 0 execution mode, but leaving the host system completely isolated and protected.

AMD-V: AMD Virtualization (AMD-V) is a set of extensions to AMD's x86 processors designed to improve the performance of virtualized environments. These, similarly, those from Intel, also give VMs direct access to HW to improve performance. These provide mechanisms to create and manage VMs that are isolated from the host OS and other VMs running on the same HW, thus giving an additional layer of protection.

Occasionally, HW-assisted CPU virtualization may not have performance benefits over the software-assisted approach. Certain events (e.g., page fault) may require frequent switching between the VM and the hypervisor for control over resources. The overhead of the VM exit, may lead sometimes it cannot outperform DBT [18].

2) Memory Virtualization

Memory virtualization plays a crucial role in creating VMs that share the host's physical memory. By abstracting physical memory, data between memories are protected, providing VMs with an isolated and secure environment [19,20].

Virtual memory virtualization adds overhead by including an extra level of indirection. A two-stage mapping process must be maintained by the guest OS and the hypervisor, as in Fig. 6. Virtual memory (blue) needs to be translated into "physical" memory of the guest OS (gray), and then into host physical memory (green) in an efficient manner.

Shadow Paging, Intel EPT and AMD NPT: Two of the main categories of memory virtualization techniques are the software-assisted ones that uses Shadow Paging or Shadow Page Tables (SPT), and the hardware-assisted, known as SLAT (Second Level Address Translation (SLAT)) or nested paging, such as Intel EPT (Extended Page Tables) or AMD NPT (Nested Page Tables) for x86 [21-24].

In Shadow paging, the hypervisor walks the Guest Page Tables and creates the SPTs, although after the establishment of the SPTs the memory accesses are very fast. In HW-assisted, it's the HW that walks the Guest and Host Page Tables directly, not involving the hypervisor and simplifying its development. However, performance benefits may be lost with heavy TLB failures, involving many memory accesses to complete the walk [24].

Other less comprehensive techniques are Memory overcommitment, and Direct paging (para-virtualization). However, Direct paging is poorly supported by OS.

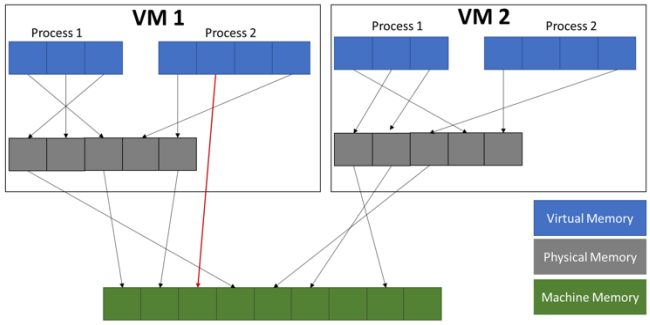

Fig. 6. Two-stage mapping process.

3) I/O Virtualization

I/O device virtualization consists of building a virtual version of the device, and then virtualizing the I/O activity directed at the device [6,25]. The hypervisor is usually directly involved in every I/O transaction. As a result, it is easy to become bottlenecked. I/O virtualization performance is critical for some applications, such as in VM-based HPC applications [26]. There are 3 main forms of I/O virtualization [6]:

Full I/O Virtualization: All functions of the device or BUS infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated by software. This software resides in the hypervisor and acts as a virtual device. A single hardware device can be shared by multiple VMs simultaneously. However, this form of virtualization can require significant HW resources. By involving the hypervisor in each I/O operation, latency can increase, and throughput rates may be lower.

I/O Paravirtualization or Split Driver Model: A frontend driver runs in Domain U (guest domain) and a backend driver in Domain 0 (privileged domain) to virtualize the I/O devices. Domain 0 directly accesses the I/O devices, and exports SGD (Simplified Generic Devices) generic devices to Domain U, while the frontend driver manages the guest OS's I/O requests. For example, a guest OS views a disk as a GBD (Generic Block Device), while Domain 0 executes the drivers of real disk. Paravirtualization offers better performance than full virtualization, but at the cost of higher processor overhead.

I/O Direct Virtualization: This type of virtualization leaves the VM accessing the devices directly. It can achieve a near-native performance experience without processor overhead. Hardware assistance and standards were developed to target these problems, including Intel Virtualization Technology for Directed I/O (VT-d), AMD I/O Virtualization (AMD-Vi or just IOMMU), and the PCI-SIG IOV standard [26-28].

Both VT-d and AMD-Vi allow direct access to peripheral devices such as Ethernet, and hard disks through Direct Memory Access (DMA) and interrupt remapping by the VM's guest OS with the help of an Input/Output MMU (IOMMU) that translates virtual device addresses into physical addresses. PCI-SIG IOV standard define extensions to the PCI express (PCIe) to enable VMs to share a device without other HW support.

4) Storage Virtualization

Storage virtualization is similar to certain forms of I/O virtualization in that the physical hard disk can be partitioned into multiple virtual disks that are made available to VMs [6]. A virtual disk is treated by the VM as a physical disk. To virtualize an I/O request to the virtual disk, the hypervisor needs to map the parameters (e.g., the track and sector locations) onto the corresponding parameters of the physical disk, and using this mapping send the request to the disk controller. Modern hypervisors (e.g., Hyper-V, VMware vSphere or XEN) to achieve HA can also use shared storage areas to minimize downtime and business impact [29]. Both physical SAN and NAS as well as virtualized SANs can be used for this purpose.

E. Virtualization Security

Virtualization offers numerous benefits, but it also introduces various security vulnerabilities and threats. These can be categorized into different families, such as: hypervisor vulnerabilities, virtualization detection, VM-based rootkits or hyperjacking, I/O channel attacks, and side channel/covert channel attacks [29,30].

Exploiting these vulnerabilities may enable hackers to compromise hypervisors, obtain privileges and take control of host resources. Cross-VM, VM escape, VM Sprawl, VM hopping, DoS of host and Data theft are examples of common attacks [31].

While regular security measures, such as install hypervisor updates, harden the VMs and the hypervisor management system are important, they may be insufficient against certain sophisticated attacks, such as VM-based rootkits and hyperjacking. Additional measures based on advanced hardware-assistance are crucial, including hardware-assisted secure launching of the hypervisor and scanning hardware-level details to assess the integrity of the hypervisor and identify rogue hypervisors. Current HW security extension, such as Intel (TXT,SGX), and AMD SVM, play thus a critical role in achieving platform security, by providing a trusted execution environment [32,33]. However, advances are needed given existing architectures typically support only one secure execution environment.

III. SCOPE AND RELATED WORK

The primary focus of this research is to study system VMs and the hypervisor-based virtualization supporting them. Specifically, this research aims to assess their maturity thrurough the performance evaluation of modern hypervisors (type I and type II), which play a critical role in the design of VMM layer.

Virtualization techniques face performance challenges due to the added overhead compared to the native environment. Initially, systems VMs had significant overhead, but this has been reduced over the years due to software optimizations and hardware advances.

Recognizing this, extensive performance evaluations of hypervisors and non-virtualized execution have been conducted [34-38]. Nevertheless, previous comparison of VMs mostly relied on older software like Xen, KVM, or even VMware and out-of-tree patches. In addition, there is a lack of evidence these studies have fully utilized new HW extensions, particularly Intel-VT/AMD-V. This is especially relevant for type-II hypervisors, which historically were not able to take advantage of these extensions, as they are nowadays.

This paper specifically describes the performance of system VMs and considers two modern type I and type II hypervisors, Hyper-V and Virtual Box, for evaluation. These types are well-supported as they do not require OS modifications and hypervisors are configured to fully utilize advantage of current HW features. The evaluation will primarily assess the added overhead on a resource-by-resource basis.

While each virtualization technique may introduce its own set of performance implications, this study aims to provide an overall evaluation of the maturity of this form of virtualization, rather than focusing on specific problems or mechanisms. For instance, at the CPU level, virtualization may encounter issues such as Double Scheduling, Scheduling Fairness, or Asymmetric CPUs. Memory-level challenges could involve Memory Reclamation or Memory Duplication, while disk-related problems may include I/O Scheduling or Layered Filesystems. In the network domain, virtualization might face obstacles in packet processing due to NAT mechanisms or network instability.

Other forms of virtualization, such as Container-based virtualization [39,40], which present themselves as an interesting alternative to VMs in virtual datacenters or cloud will not be the subject of study or evaluation in this paper. Also, neither codesigned nor process-level VMs are target.

Finally, addressing virtualization security challenges is out of the scope of this work.

IV. EXPERIMENTAL SET-UP AND TESTING METHODOLOGY

This section gives an overview of performance testing, including host systems, testing tools used, and the methodologies and metrics for evaluation.

A. Host Systems, Virtualization and Benchmarking Tools

In this subsection, the characteristics of the host systems and as well as the benchmark tools employed are presented.

1) Host Systems and Configurations

A set of hardware representative platforms was chosen with different CPU, memory configurations, disks and network interfaces enabling analysis of the support for several virtualized environments:

(1) AMD Laptop

- *CPU: AMD Ryzen 5 3550H*
- *RAM: 8GB (1x8GB) DDR4 2400Mhz CL16*
- *Storage: SSD NVMe Gen3x4 Micron 2200 500GB*
- *Network interface: Realtek RTL8168 Gigabit*

(2) Intel Laptop
- *CPU: Intel Core I7 6700HQ*
- *RAM: 8GB (1x8GB) DDR4 2133Mhz CL16*
- *Storage: HDD SATA 3 Hitachi 7200RPM 1TB*
- *Network interface: Killer E2400 Gigabit*

(3) AMD Desktop
- *CPU: AMD Ryzen 7 5900x @5.0Ghz*
- *RAM: 16GB (2x8GB) DDR4 3133Mhz CL15*
- *Storage: SSD NVMe Gen4x4 Samsung 980 Pro 1TB*
- *Network interface: Realtek 8111H Gigabit*

2) Operating Systems

The host OS is chosen based on the hardware compatibility and virtualization support, while guest OS is chosen depending on the application to be virtualized. OSs that were chosen for the tests were the following:

Windows 10 [41]: Windows 10, already replaced by Windows 11, is a widely used OS for personal use and compatible with various virtualization platforms such as VMware Workstation, Oracle VirtualBox, Hyper-V among others, making it one of the preferred OS in virtualization and testing environments.

Garuda Linux Xfce [42]: An arch based, Linux distribution was selected as it is one of the preferred OS for virtual DCs and Cloud. The reasons why the Garuda Linux Xfce was chosen for virtualization were threefold, its performance, its stable and lightweight desktop environment, and its extensive driver/software support.

3) Virtualization Tools

From the previously presented options, one hypervisor of each type (type 1 and type 2) was selected to be part of our test suite for creating system-level VMs, these were:

Oracle VirtualBox [43]: VirtualBox is a free, open source, Type II virtualization platform developed by Oracle. VirtualBox is a popular choice for home users and small businesses because it is easy to use and offers advanced features and supports different OSes.

Hyper-V [44]: Hyper-V is one of the proprietary Type I virtualization platforms developed by Microsoft for Windows OS. Hyper-V can be used by home and business users, but is exclusive to the Pro, Enterprise and Server versions of Windows.

4) Benchmarks

Several testing software tools or utilities, also known as *benchmarks*, enable the analysis and performance comparison between different systems. Table 1 provides examples of benchmarks categorized by resource under test.

TABLE 1. BENCHMARK TOOLS

Resource	*Benchmarks*
CPU	Passmark Geekbench Cinebench Novabench
Memory	MemTest86 Memtest64 AIDA64
I/O Disk	CrystalDiskMark iOZone
Network	iPerf NetPerf

The selected benchmarking tools must be cross-platform, compatible with the target virtualization technologies, and support the metrics tests listed in Table 1. The chosen tools or software utilities for the benchmarks were the following:

GeekBench [45]: Geekbench is a cross-platform CPU benchmarking utility regularly updated to ensure compatibility and evolving user needs. It runs a set of CPU tests to assess different aspects of CPU performance. Two CPU performance values are returned: a Single-Core Score and a Multi-Core Score.

PassMark [46]: PassMark is a software utility for performing benchmark tests on a computer system. The software can perform CPU stress tests, 2D and 3D graphics tests, hard disk tests, and memory tests. Like the GeekBench tool, scores are generated depending on the tests performed.

CrystalDiskMark [47]: This tool tests the performance of storage drives, such as hard drives and USB flash drives. It performs read and write tests with different block sizes and conditions. *Sequential read/write* tests can be performed to measure the maximum read and write speed the unit can achieve, or *random read/write* tests.

iPerf3 [48]: A tool used to perform network tests, which can create synthetic TCP data streams and UDP datagram streams to measure the network throughput. Iperf3 allows setting various parameters to test or optimize a network.

B. Summary of Testing Methodology and Data Collection

To compare the overhead of hypervisors with native systems and ensure rigorous analysis, we maintained consistent configurations throughout the experiments. Details on the experimental configurations, tools, and methodologies employed are provided next.

The host systems used for testing had the Oracle VirtualBox hypervisor installed, and Hyper-V was activated. Additionally, the virtualization extensions (Intel VT-d or AMD SVM) were enabled in the BIOS. These systems were running Windows 10 22H2 and Garuda Linux Talon (Xfce). To avoid biasing the results, all software, including programs, applications, and drivers, was updated to their latest versions to ensure proper functioning. To make the results comparable between the virtualized operating systems, Windows 10 and Garuda Linux Talon (Xfce), the systems were limited to 4 threads, allowing for a direct comparison with the virtualized systems.

The GeekBench, PassMark, CrystalDiskMark and iperf3 benchmark tools were used to generate different workloads and measure different performance metrics. In PassMark testing, only CPU and RAM tests were run as they are the only common tests to both Windows and Linux SO. In CrystalDiskMark was used the Q8T1 Sequential Write test and the Q32T1 4K Random Read test, as they best represent daily OS usage. The *File Size* was increased to 2GB, in order to surpass the AMD desktop system SSD 1GB cache. Note that, AMD systems have SSD disk drives, and the Intel system has an HDD disk drive. For the network throughput measurement with iperf3, an internal network was setup supported by CAT6 cables, and a MEO GR141DG router.

All tests were run five times, separated by a two-minute waiting period, to ensure consistent and reliable results. The data was then processed using geometric averaging to mitigate potential interpretation errors and minimize bias in the results [49].

V. EXPERIMENTAL RESULTS

This section presents benchmark results comparing the hypervisor overhead to native execution. We will analyze and discuss the result of each benchmark using workload metrics like GeekBench and PassMark CPU and memory scores, disk read/write speeds, network packet losses and throughput to determine the overhead of virtualization.

A. *Native Systems*

First the results of the benchmarks in a native run, without any restrictions, are presented.

1) *GeekBench Results*

Fig. 7 and Fig. 8 show the results of the GeekBench tests. In both tests, it can be observed that the change of OS from Windows to Linux leads to a slight performance increase for all analyzed host platforms. It can also be seen that the Intel-based laptop and the AMD-based desktop achieved a significant performance increase in the multi-core tests, with the Intel laptop having an increase of 29% and the desktop having an increase of 7%.

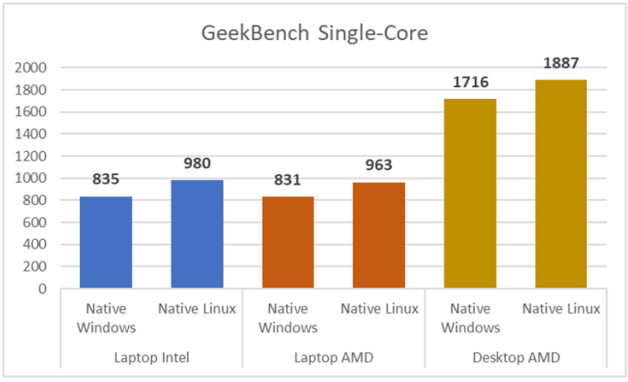

Fig. 7. GeekBench Single-Core comparison (higher scores is better).

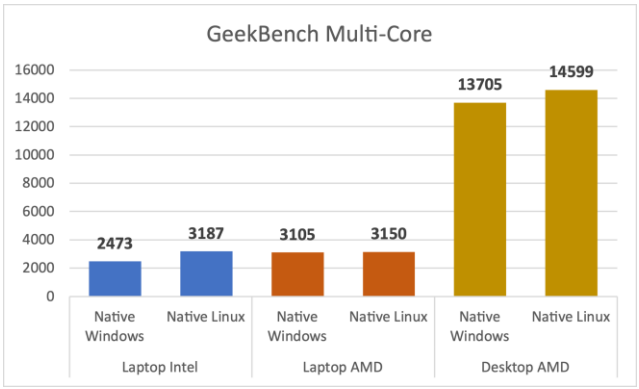

Fig. 8. GeekBench Multi-Core comparison (higher scores is better).

2) *Passmark Results*

Fig. 9 and Fig. 10 presents the results of the CPU and memory tests with PassMark. In the CPU test, a barely noticeable negative trend was verified 4% in the AMD systems, while in the Intel system was observed a minimal increase of 1% in the score obtained. Regarding the memory tests, a barely significant increasing trend was verified on all machines of only 1%, being more substantial of 7% on the desktop AMD system.

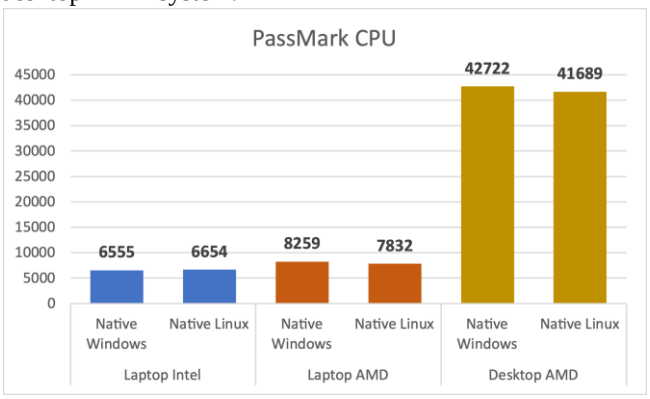

Fig. 9. PassMark CPU comparison (higher scores is better).

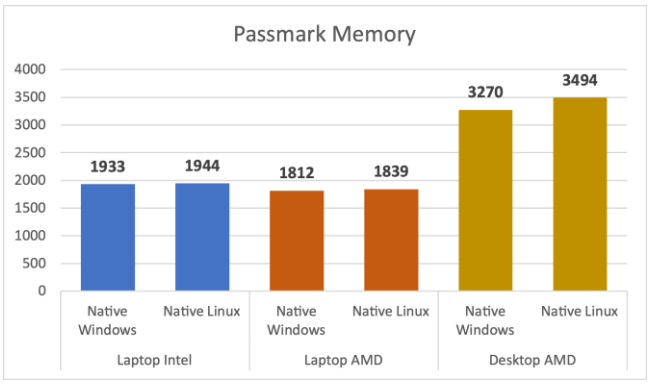

Fig. 10. PassMark memory comparison (higher scores is better).

B. *Virtualized Systems*

This section presents the results of the benchmarks conducted on a native Windows 10 system, and on the virtualized systems using both VirtualBox and Hyper-V.

1) *Limitations of Virtualized Systems*

Oracle Virtual Box limits the number of threads assigned to each VM to half of the total threads of the host system. Since both laptops used have 8 threads, this limit is 4 threads. To make results directly comparable, all environments were limited to 4 threads and 3GB of RAM.

2) *GeekBench Results*

Native System: Fig. 11 compares GeekBench natively without any restrictions to GeekBench limited to 4 Threads. Both laptops were expected to show a 50% performance drop, due to the loss of half of their threads (from 8 to 4). However, the performance loss was around 30% to 40%, indicating that GeekBench may not be completely scalable with respect to the number of threads. On the desktop AMD system, the performance loss should also be proportional to the loss of threads from 24 to 4, which equates to a loss of 83%. However, the performance loss was only 72%.

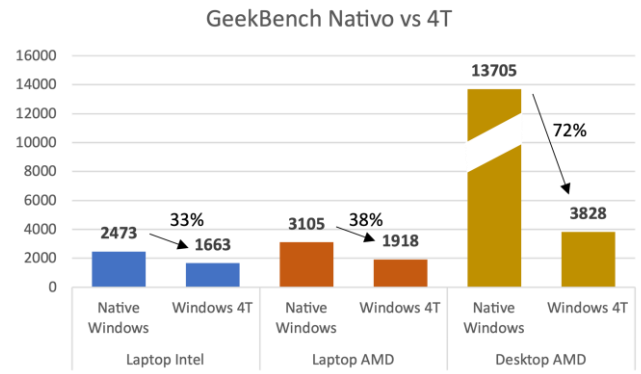

Fig. 11. Geekbench native vs 4 threads comparison (higher scores is better).

Single-Thread: The results of the GeekBench Single-Threaded tests are presented in Fig. 12. The single-threaded performance tests show two trends. First, the performance of Linux virtualized using Oracle VirtualBox is similar to that of native Windows, with AMD systems showing a slight drop in performance. Second, all systems perform significantly better in the virtualized environment using Hyper-V, ranging from 5% on the AMD desktop to 15% on the Intel laptop.

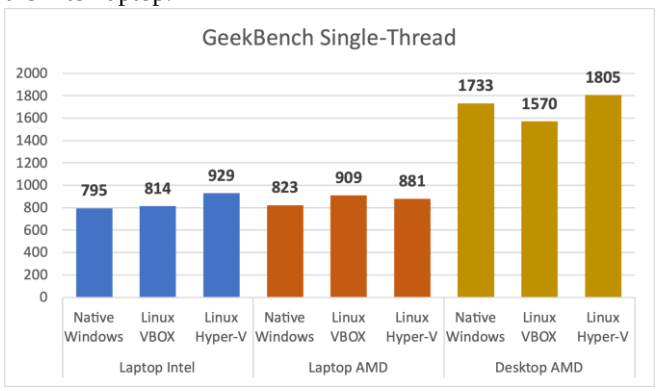

Fig. 12. GeekBench Single-Thread comparison (higher scores is better).

4 Thread: Finally, Fig. 13 presents the results of the GeekBench tests with 4 Threads. These show that with 4 threads the AMD-based systems achieved the best performance using the Virtual Box environment, contrary to the results obtained earlier in the single threaded tests, the Intel system, as before, achieved better performance using the Hyper-V environment.

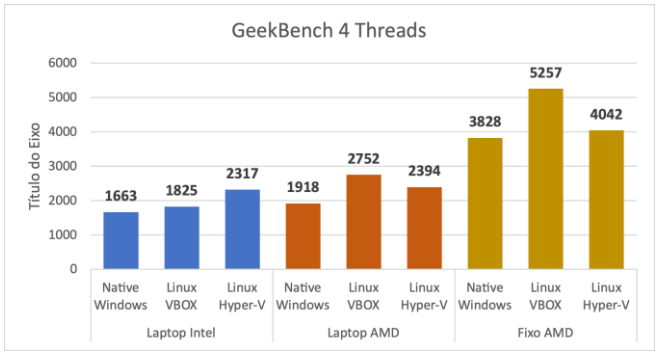

Fig. 13. GeekBench 4 Thread comparison (higher scores is better).

3) Passmark Results

Fig. 14 and Fig. 15 show the results of the CPU and memory evaluation tests using PassMark.

CPU: The results obtained showed the same trend as the GeekBench, 4 thread results, where the AMD systems perform better on Virtual Box, while the Intel system performs better on Hyper-V.

Memory: Memory tests showed that regardless of the system, there is a performance loss when using virtualization. Laptops showed a smaller loss, although the Intel laptop can partially recover performance using Linux Hyper-V virtualization.

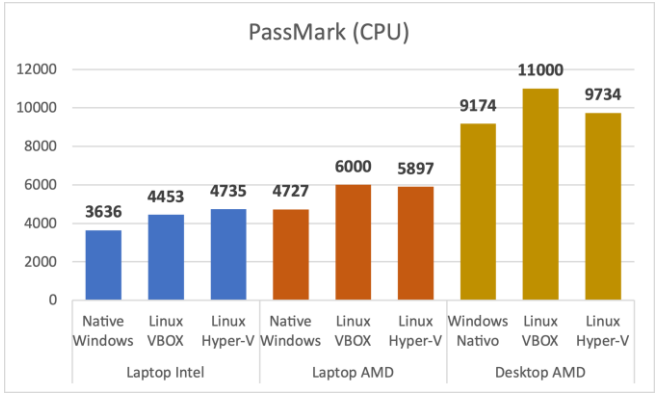

Fig. 14. PassMark CPU comparison (higher scores is better).

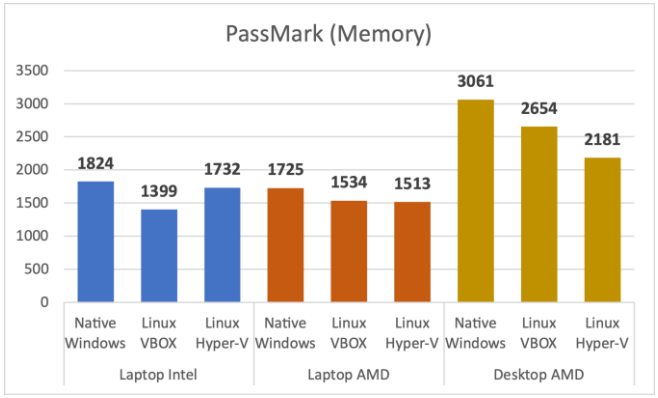

Fig. 15. PassMark Memory comparison (higher scores is better).

4) CrystalDiskMark Results

Fig. 16 and Fig. 17 present the results of the disk write and read operations tests in MB/s using CrystalDiskMark.

SEQ Write Q8T1: The results show Native Windows had the best overall performance, followed by Native Linux and then Linux Hyper-V. On the other hand, Linux Virtual Box showed the worst performance, with significant losses of up to 66% in write speed compared to Native Windows.

4K Random Read Q32T1: The analysis of the results showed that the Native Windows OS had the best overall performance. In second place is OS Native Linux, followed by Linux Hyper-V. The worst performance was observed in Linux Virtual Box, with drastic losses of up to 92% on the desktop AMD system.

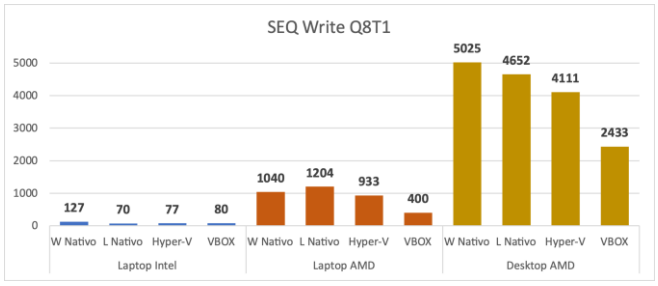

Fig. 16. CrystalDiskMark SEQ Write Q8T1 comparison (MB/s).

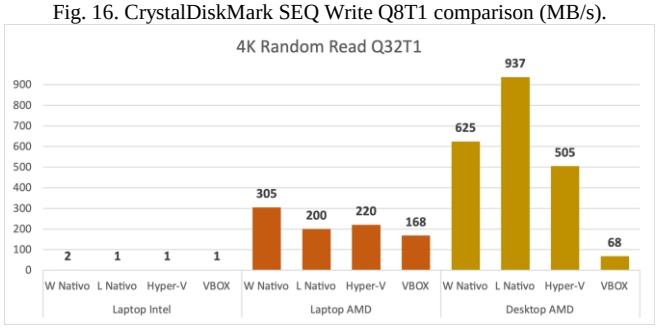

Fig. 17. CrystalDiskMark 4K Random Read Q32T1 comparison (MB/s)

5) iPerf3 Results

Finally, the results of network tests on TCP and UDP communications using the iPerf3 tool are presented in Fig. 18 to Fig. 20.

TCP Bandwidth: All systems demonstrated ability to efficiently handle TCP streams. The Intel system showed a worse scenario in both virtualized environments but continues to perform decently in terms of TCP packet bandwidth.

UDP Bandwidth: All systems were able to handle UDP packet transfer efficiently in terms of bandwidth. The Intel host system, despite the lower values, still performs well UDP in terms of bandwidth.

UDP Packet Loss: Since the UDP protocol does not use handshakes in communication, meaning that there is no packet delivery verification, which can result in packet loss in transmission. During testing, the Intel system showed the highest number of lost packets. However, it did not experience any problems using the Linux natively. The AMD laptop showed negligible packet loss.

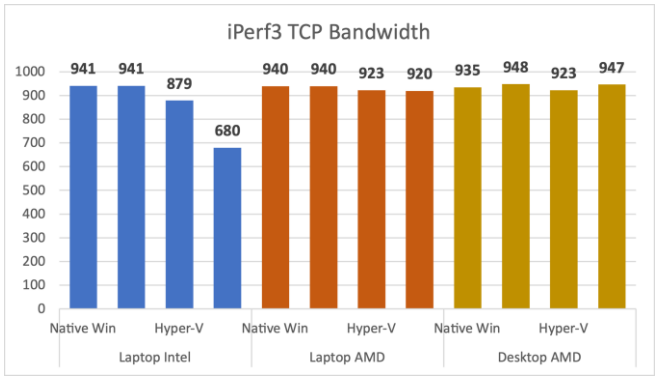

Fig. 18. iPerf3 TCP Bandwidth comparison (Mb/s).

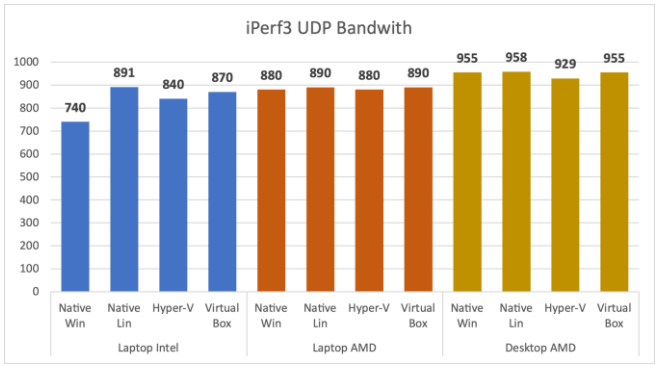

Fig. 19. iPerf3 UDP Bandwidth comparison (Mb/s).

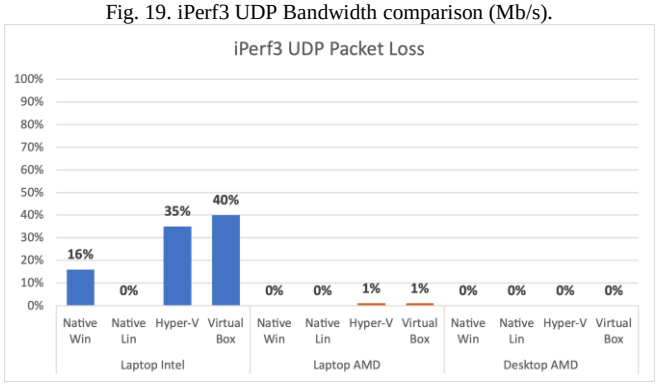

Fig. 20. iPerf3 UDP Packet Loss comparison (%).

C. Discussion

The study conducted performance evaluations on virtualization techniques, focusing on the comparison between native and virtualized environments.

The measured benchmarking metrics revealed an increase in performance degradation, manifesting the presence of overhead associated with virtualization. However, the results obtained from comparing different hypervisors, specifically Hyper-V (type I) and Virtual Box (type II), showed that both types tend to impose minimal overhead on CPU, memory, disk, and network tests. Despite their different natures (Hyper-V being a native OS feature and Virtual Box being a software installed on top of the OS), both hypervisors effectively manage to take advantage of HW extensions.

An interesting observation, albeit relatively minor and not within the main scope of this paper, is about the choice of virtualization solution based on the CPU model. In the GeekBench/PassMark test pair, evaluating CPU and memory performance, systems with AMD CPUs performed better in the Virtual Box environment (type II), while systems with Intel CPUs demonstrated better performance in the virtualization tests using the Hyper-V (type I).

In contrast, the results obtained from the CrystalDiskMark/iPerf3 test pair, which analyzed storage and network throughput, clearly aligned with the available hardware resources. Systems with lower hardware resources experienced diminished performance in both Hyper-V (type I) and Virtual Box (type II) virtualization scenarios.

VI. CONCLUSIONS

Hypervisor-based virtualization is a technology that has evolved since the 1970s and is widely used by end users and in the creation of virtual data centers and cloud infrastructures.

To assess its maturity, we conducted a set of performance tests using a modern set of hardware systems, operating systems, hypervisors, and benchmark tools, with a focus on type I versus type II hypervisors, while using proper workload metrics. We did not compare para-virtualization as it requires modifications to the operating systems and is not widely used.

Our evaluations showed that, as expected, native environments outperformed virtualized ones. The results also showed that virtualization still adds some overhead, though the performance penalties for CPU, memory, disk, and network are not significant.

Furthermore, no relevant differences were observed between types I and II hypervisors (Hyper-V vs Virtual Box), which is noteworthy considering the history of Type II hypervisors. Modern type II hypervisors also are taking advantage of the potential of HW components like type I hypervisors.

In summary, our results have attested and confirmed the maturity of hypervisor-based virtualization technology.

REFERENCES

[1] O. AbdElRahem, A. Bahaa-Eldin, and A. Taha, "Virtualization security: A survey," *2016 11th International Conference on Computer Engineering & Systems (ICCES)*, Cairo, Egypt. 2016, pp. 32-40. DOI: https://doi.org/10.1109/ICCES.2016.7821971

[2] D. C. Marinescu, "Cloud Resource Virtualization," *Cloud Computing*, 2º ed., Elsevier, 2018, pp. 365-402.

[3] E. Bugnion, S. Devine, M. Rosenblum, J. Sugerman, and E. Wang, "Bringing Virtualization to the x86 Architecture with the Original VMware Workstation," *ACM Transactions on Computer Systems*, Volume 30, Issue 4, Article No.: 12pp 1–51, November 2012. DOI: https://doi.org/10.1145/2382553.2382554

[4] A. Varasteh and M. Goudarzi, "Server Consolidation Techniques in Virtualized Data Centers: A Survey," *IEEE Systems Journal*, vol. 11, no. 2, pp.772-783, June 2017. DOI: 10.1109/JSYST.2015.2458273.

[5] M.J. Kavis, *Cloud Service Models. Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*, Wiley, Hoboken, NJ, USA, 2014.

[6] K. Hwang, G. Fox and J. Dongarra, *Distributed and cloud computing: from parallel processing to the Internet of things*, 1st Ed., Morgan Kaufmann, 2011.

[7] D. Chisnall, *The definitive guide to the Xen hypervisor*, Pearson Open Source Software Development Series, 1st Ed., Pearson Education, Boston, MA, USA, 2008.

[8] J. Popek and P. Goldberg, "Formal requirements for virtualizable third generation architectures", *Communications of the ACM, 17*(7), 412-421, 1974. DOI: https://doi. org/10.1145/361011.361073

[9] F. Mehdipour, H. Noori e B. Javadi, "Energy-Efficient Big Data Analytics in Datacenters," *Advances in Computers*, vol. 100, Elsevier, 2016, pp. 59-101.

[10] J. D. Gelas, "Hardware Virtualization: the Nuts and Bolts," AnandTech, March 2008. [Online]. Available: https://www.anandtech.com/show/2480/09

[11] L. Sepúlveda-Rodríguez, J. Chavarro-Porras, J. Sanabria-Ordóñez, H. Castro, J. Matthews, "A Survey of Virtualization Technologies: Towards a New Taxonomic Proposal," *Ingeniería e Investigación*, 42(3), 2022. DOI: https://doi.org/10.15446/ing.investig.97363

[12] VMware white paper, "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", 2008 [Online]. Available: https://www.vmware.com/techpapers/2007/understanding-full-virtualization-paravirtualizat-1008.html

[13] S. A. Babu, M. J. Hareesh, J. P. Martin, S. Cherian e Y. Sastri, "System Performance Evaluation of Para Virtualization, Container Virtualization, and Full Virtualization Using Xen, OpenVZ, and XenServer," *International Conference on Advances in Computing and Communications (ICACC)*, Cochin, India, 2014.

[14] *Intel 64 IA-32*, "Intel® 64 and IA-32 Architectures Software Developer Manuals," March 2020. [Online]. Available: http://software.intel.com/en-us/articles/intel-sdm.

[15] *ARM ISA*, "Instruction Set Architecture (ISA)," ARM, [Online]. Available: https://www.arm.com/glossary/isa

[16] *Intel-VT*, "Intel® Virtualization Technology List," November 2011. [Online]. Available: https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html

[17] *Qemu*, "Qemu process emulator," [Online] Available: https://www.qemu.org

[18] X. Ding and J. Shan, "Diagnosing Virtualization Overhead for Multi-threaded Computation on Multicore Platforms," *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, Vancouver, BC, Canada, 2015, pp. 226-233. DOI: 10.1109/CloudCom.2015.102.

[19] VMWare, "Memory Virtualization Basics," May 2019. [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/8.0/vsphere-resource-management/GUID-9D2D0E45-D741-476F-8DB1-F737839C2108.html

[20] VMWare, "Software-Based Memory Virtualization," May 2019. [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.resmgmt.doc/GUID-7BB72010-029B-46F4-B6BF-6277E129B940.html.

[21] Y. Tai, W. Cai, Q. Liu, G. Zhang and W. Wang, "Comparisons of memory virtualization solutions for architectures with software-managed TLBs," *International Conference on Networking, Architecture, and Storage (NAS)*, Xi'an, China, 2013.

[22] VMware white paper, "Performance Evaluation of Intel EPT Hardware Assist," 2009. [Online]. Available: https://www.vmware.com/pdf/Perf_ESX_Intel-EPT-eval.pdf

[23] *AMD-V*, "AMD-V™ Nested Paging," July 2008. [Online]. Available: http://developer.amd.com/wordpress/media/2012/10/NPT-WP-1%201-final-TM.pdf.

[24] J. Gandhi, A. Basu, M. D. Hill and M. M. Swift, "Efficient Memory Virtualization: Reducing Dimensionality of Nested Page Walks," *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Cambridge, UK, 2014.

[25] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauery, I. Pratt, and A. Wareld, Xen and the Art of Virtualization, *19th ACM Symposium on Operating System Principles (SOSP'03)*. ACM, 2003.

[26] D. Muench, O. Isfort, K. Mueller, M. Paulitsch e A. Herkersdorf, "Hardware-Based I/O Virtualization for Mixed Criticality Real-Time Systems Using PCIe SR-IOV," *IEEE International Conference on Computational Science and Engineering*, CSE, Sydney, NSW, Australia, 2013.

[27] *AMD IOMMU*, "AMD I/O Virtualization Technology (IOMMU) Specification," February 2009. [Online]. Available: https://developer.amd.com/wordpress/media/2012/10/34434-IOMMU-Rev_1.26_2-11-09.pdf.

[28] *Intel-VT I/O*, "Intel® Virtualization Technology for Directed I/O," October 2014. [Online]. Available: https://lettieri.iet.unipi.it/virtualization/vt-directed-io-spec.pdf.

[29] J. Hoopes, *Virtualization for Security*, 1st ed., Syngress, 2009.

[30] B. Asvija, R. Eswari, M.B. Bijoy, "Security in hardware assisted virtualization for cloud computing-State of the art issues and challenges," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Volume 151, Issue C, Marc 2019, pp 68-92. DOI: https://doi.org/10.1016/j.comnet.2019.01.013

[31] M. Compastié, R. Badonnel, O. Festor and R. He, "From virtualization security issues to cloud protection opportunities: An in-depth analysis of system virtualization models," *Computers & Security*, Volume 97, 2020. DOI: https://doi.org/10.1016/j.cose.2020.101905.

[32] *Intel TXT*, "Intel Trusted Execution Technology Measured Launched Environment Programming Guide". [Online]. Available: http://www.intel.eu/content/www/eu/en/software-developers/intel-txt-software-development-guide.html

[33] *AMD SVM*, "Secure Virtual Machine Reference Manual," Maio 2005. [Online]. Available: https://markowsky.us/papers/amd-docs/amd-pacifica-specification.pdf

[34] J. Hwang, S. Zeng, F. Wu, and T. Wood. "A component-based performance comparison of four hypervisors", *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pages 269–276,

[35] P. Swati, and S. Singh, "Performance comparison of VMware and Xen hypervisor on Guest OS," *International Journal of Innovative Computer Science & Engineering (IJICSE)*, Vol. 2, Issue 3, 2015, pp 56-60.

[36] D. Vojnak, B. Đorđević, V. Timčenko and S. Štrbac, "Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation," *2019 27th Telecommunications Forum (TELFOR*), Serbia, 2019, pp. 1-4. DOI: 10.1109/TELFOR48224.2019.8971213.

[37] B. Đorđević, V. Timčenko, D. Sakić and N. Davidović, "File system performance for type-1 hypervisors on the Xen and VMware ESXi," *2022 21st International Symposium INFOTEH-JAHORINA*, Bosnia and Herzegovina, 2022, pp. 1-6. DOI: 10.1109/INFOTEH53737.2022.9751288.

[38] H. Rahman et al, "Performance Evaluation of Hypervisors and the Effect of Virtual CPU on Performance," *2018 IEEE International Conference on Intelligent Computing and Communication for Smart World (I2C2SW0)*, Guangzhou, China, 2018, pp. 772-779. DOI: 10.1109/SmartWorld.2018.00146.

[39] A.K Yadav and M. Garg, "Docker containers versus virtual machine-based virtualization", *Springer International Conference on Emerging Technologies in Data Mining and Information Security*, Singapore, 2019, pp. 141–150.

[40] A. Bhardwaj, and C. Krishna, "Virtualization in Cloud Computing: Moving from Hypervisor to Containerization—A Survey", *Arabian Journal for Science and Engineering* 46, 8585–8601, 2021. DOI: https://doi.org/10.1007/s13369-021-05553-3

[41] Microsoft Inc., "Introduction to Windows 10," Microsoft, February 2021. [Online]. Available: https://web.archive.org/web/20210203032518/http://www.microsoft.com/en-gb/windows/.

[42] Garuda, "Introduction to Garuda Linux," Garuda, [Online]. Available: https://garudalinux.org/index.html

[43] VirtualBox, "Welcome to VirtualBox," Oracle, [Online]. Available: https://www.virtualbox.org/

[44] Microsoft Inc., "Introduction to Hyper-V on Windows 10," Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/

[45] Geekbench, "Introducing Geekbench 6," [Online]. Available: https://www.geekbench.com

[46] PassMark, "PassMark® Software" [Online]. Available: https://www.passmark.com

[47] Crystalmark [Online]. Available: https://crystalmark.info

[48] Iperf, "iPerf - The ultimate speed test tool for TCP, UDP and SCTP," [Online]. Available: https://iperf.fr

[49] P. Fleming e J. Wallace, "How not to lie with statistics: the correct way to summarize benchmark results," *Communications of the ACM*, vol. 29, no. 3, pp. 218-221, 1986.