File System Performance Comparison in Full Hardware Virtualization with ESXi, KVM, Hyper-V and Xen Hypervisors

Borislav DJORDJEVIC[1,2], Valentina TIMCENKO[1], Nenad KRALJEVIC[2], Nemanja MACEK[2]
[1]*Mihajlo Pupin Institute & School of Electrical Engineering, University of Belgrade, Belgrade, Serbia*
[2]*VISER, School of Electrical and Computer Engineering of Applied Studies, Belgrade, Serbia*
borislav.djordjevic@pupin.rs

***Abstract*—This paper focus is the mathematical modeling of the file system performance in virtual environment when using type-1 hypervisors. The modeling provides a set of hypotheses related to the expected behavior. The presented model is validated based on the analysis of a collection of the results obtained for a specific case study. Our case study includes the file system performance comparison, in full hardware virtualization, when examining four dominant type-1 hypervisors: ESXi, KVM, Hyper-V, and Xen. We chose Filebench as a benchmark tool, which guarantees comprehensive and versatile testing of file system performance, whereas for all tested hypervisors we have provided an equivalent environment and testing conditions. For all the examined hypervisors, we have tested the cases with one, two, and three virtual machines that are running simultaneously, whereas CentOS 6.3 Linux is used as the guest operating system. We have further validated the mathematical model and defined hypotheses by the means of the case study benchmark results.**

***Index Terms*—file systems, operating systems, performance evaluation, platform virtualization, virtual machine monitors.**

I. INTRODUCTION

In the modern age, ICT (Information and Communication Technologies) are becoming increasingly important factors, whereas virtualization techniques are constantly taking up new fields and applications.

Virtualization enables ICT resource partitioning, efficient utilization of resources, powerful system management, higher availability and better fault tolerance, all combined with stronger security. The major benefits of virtualization correspond to the cost reduction, the efficient use of ICT resources and the reduction of electricity consumption. As it contributes to environmental conservation, virtualization is considered as representative of the green technologies [1-2].

There is a number of techniques for virtualization, whereas the choice depends on the specific needs of the user. Some virtualization techniques achieve better performance, whereas others achieve greater flexibility. Actually, virtualization can be applied over many different computer components, thus there are techniques for hardware, software, desktop, database, network, RAM memory and storage virtualization.

Virtualization includes the abstraction and encapsulation of computer resources, where these resources can be used in the appropriate way for a particular application. The hardware virtualization relies on the usage of a hypervisor. A hypervisor is a software layer that functions as an intermediary between the host operating system (host OS, hOS) and VMs (virtual machines), and creates a simulated environment for VMs that may or may not have the same characteristics as the physical environment.

The hypervisors can be of two types. Hypervisor of type-1, the native hypervisor, runs directly on physical hardware (so-called bare metal). Examples of type-1 hypervisor are Citrix XenServer, Linux KVM, MS Windows Hyper-V and VMWare ESXi, which have been tested in this paper. Hypervisor of the type-2 runs as an application within the host OS (Operating System). Examples of type-2 hypervisor are Oracle VM Virtual Box and VMware Workstation. In practice, type-1 hypervisors have much better performance than type-2 hypervisors.

Hardware virtualization can be performed in different ways: full hardware virtualization or emulation, partial virtualization and paravirtualization. The full hardware virtualization includes software simulation of the complete hardware so that the guest operating system (guest OS, gOS) can be installed and executed without any further changes. This solution is the most elegant and easiest for use, but unfortunately provides low performance of the installed VMs. Hence, this poor performance can be mitigated by using special CPU features, Intel VT-x or AMD-V. The paravirtualization allows significantly better performance of the guest OSs i.e. through the use of the VMs, with drawback that it requires modifications of host/guest OSs.

II. RELATED WORK, OBJECTIVE AND MOTIVATION

The scope of this paper covers the performance analysis of type-1 hypervisors. It should be emphasized that such performance is one of the basic factors for achieving an adequate level of QoS (Quality of Service) in CC (Cloud Computing) environment. This problem has been discussed in the related work from various points of view. But, one very common approach and methodology relies on the comparative performance analysis of different hypervisors (mostly KVM, VMWare, Xen and Hyper-V), and is mainly based on the use of filesystem benchmark applications such as Bonnie++, Iozone, LMbench, LINPACK, HD Tune Pro and ATTO [3-8].

Some recent research related papers focus on the problem of support for appropriate I/O speeds, enabling the full range of CC applications, their effective functioning and tackle the

This work was supported by the Ministry of Education, Science and Technological Development of Republic of Serbia.

Digital Object Identifier 10.4316/AECE.2021.01002

problem of I/O performances in a virtual environment [9-12].

It has been noticed that the experimental results mainly relate to the impact of the estimated costs for realizing CC technologies [13]. It is important to choose a solution for virtual infrastructure management, as cloud resources can be quite limited and inadequate due to the need to respond to changes in a dynamic and stochastic environment [14-15]. Just a few studies provide comparative analysis of the top market virtualization hypervisors, which is highly related to our focus as well [8], [16-19].

In this paper, the main contribution is the comprehensive mathematical modeling of the FS (File System) performance in VE (Virtual Environment) when using type-1 hypervisors. The modeling of such a complex system includes many factors, which can be explored as interdependent and/or as mutually correlated. The model is applicable to most VE, and based on a model we interpret a range of practical results when targeting one particular case study.

The idea is to provide a mathematical model, apply it on a particular case study, and then provide the interpretation of a range of practical results as a validation of model.

The indicated CS (Case Study) provides the FS performance evaluation when examining the performances of four selected type-1 hypervisors, in fair play conditions. This fair competition assumes the use of identical hardware for hypervisors, the same characteristics of the generated VMs and the identical version of the guest OS. In this case, we have selected ESXi, KVM, Hyper-V, Xen, which are all using the full hardware virtualization. It is worth to note that besides full virtualization, Xen and Hyper-V can also support the paravirtualization. We have used the modern benchmark, Filebench. It is a multi-threaded based benchmark, easily configurable, and adequate for the simulation of the real-world applications. We have chosen the four different test workloads, which are similar to real applications: web server, email server, fileserver, and random file access. We have first set up the mathematical model for workloads and hypervisors; then, we defined the hypotheses related to the expected behavior of the FS, and finally proceeded with the benchmark measurements. The results were interpreted on the basis of the defined mathematical model and hypotheses.

III. XEN, KVM, HYPER-V, AND ESXI

ESXi (Fig. 1) is a type-1 hypervisor, which means that it runs directly on the hardware (bare metal), and therefore it significantly improves the system performance. It is based on an OS-independent ultra-thin architecture. The fundamental parts of the ESXi architecture are VMkernel and processes that run above it. VMkernel provides startup tools for all components, including VMs, management applications, and agents.

VMkernel has control over all hardware devices on the server and manages the resources that are needed by VM [6, 20]. ESXi applies one virtualization type, the original VMware FHV (Full Hardware Virtualization).

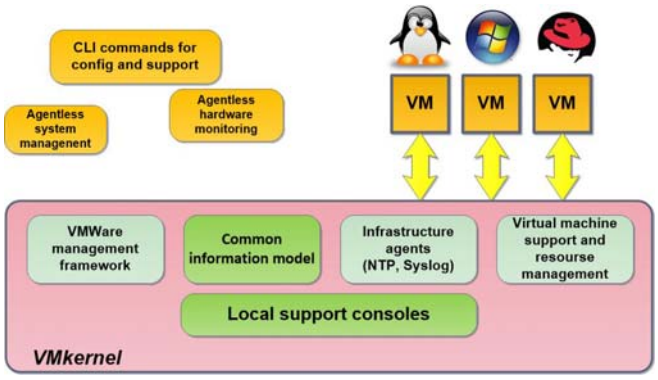

Figure 1. ESXi architecture

Xen (Fig. 2) is a native, bare-metal hypervisor, which runs directly on the hardware, allowing computer hardware to run multiple guest VMs at the same time [21-22]. Xen can work on various computer CPU architectures: x86, x86-64, Itanium, Power PC, and ARM. It currently supports the following OSs for the guest side: Linux, NetBSD, FreeBSD, Solaris, and MS Windows OS family. Xen implements two virtualization types: FHV (QEMU based) and PV (paravirtualization) which is suitable for open-source PV guests. Xen hypervisor works directly on the hardware and represents an interface for all hardware requirements such as CPU, memory and I/O for gOS. Xen uses two kinds of the domain:

- Dom0 (Domain0) – it is an integral part of Xen that operates like the real host OS. Dom0 is automatically launched after the Xen hypervisor is started. Dom0 has special privileges as it allows direct hardware access, such as the access to all the I/O devices. Dom0 provides access to hardware through real, native drivers of the hOS.
- DomU (DomainU) – it is the gOS which is launched and controlled by dom0 and works in isolation. Guests are either started with a specially modified OS by using Xen paravirtualization (PV guest) or with an unmodified but hardware-assisted OS (HVM guest).

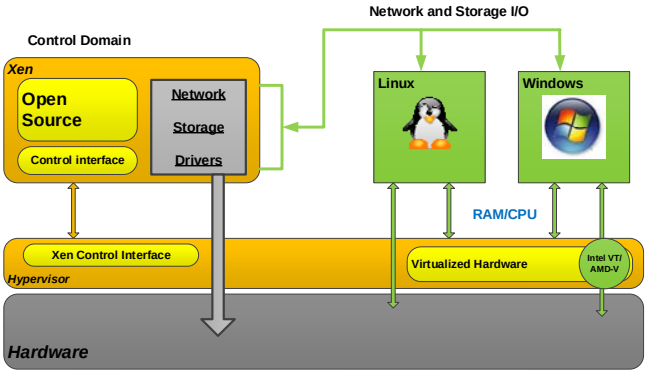

Figure 2. Xen Server architecture

KVM (Fig. 3) is a set of software, techniques, and methods that provide Linux kernel changes in hypervisor type-1 [23-24]. Using KVM, the kernel is practically gaining an extension-kernel module that allows it to run VMs. It is a part of the Linux distributions from the kernel version 2.6.20. To make the installation possible, it is required the hardware support, which consists of Intel VT-x or AMD-V add-ons, covering almost all processors that can be found in use today.

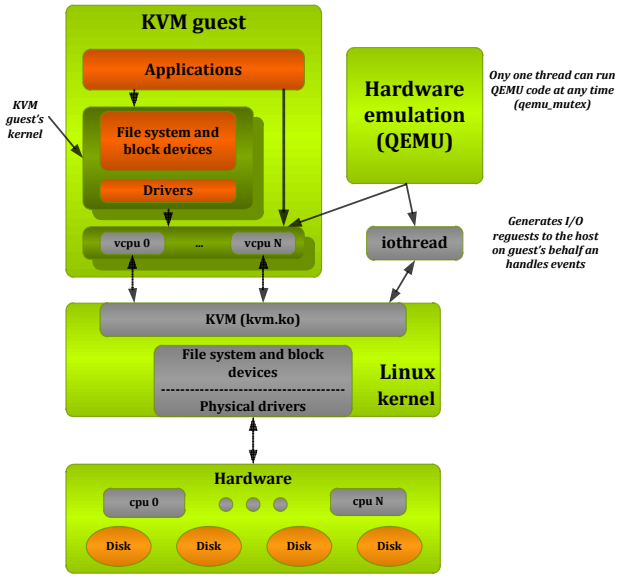

Figure 3. KVM architecture

Although initially conceived exclusively for x86 architecture, it is adapted for the use on Power-PC, IBM System/390, IA-64 and ARM platforms. The host OS can be any modern Linux distribution. KVM can be installed in an extremely simple way when installing the Linux OS. When installed, it allows the creation, management, and deletion of VMs whose maximum number is limited only by the power of the available hardware. KVM applies only the FHV (QEMU based) virtualization.

Hyper-V (Fig. 4) is type-1 (bare-metal) hypervisor-based virtualization system for x86-64 systems. It is installed on the Windows Server as any other service (Computer Browser, Application Layer Gateway, DNS Client, DHCP Client), as a role [25-26]. The hypervisor is a thin software that is located just above the hardware.

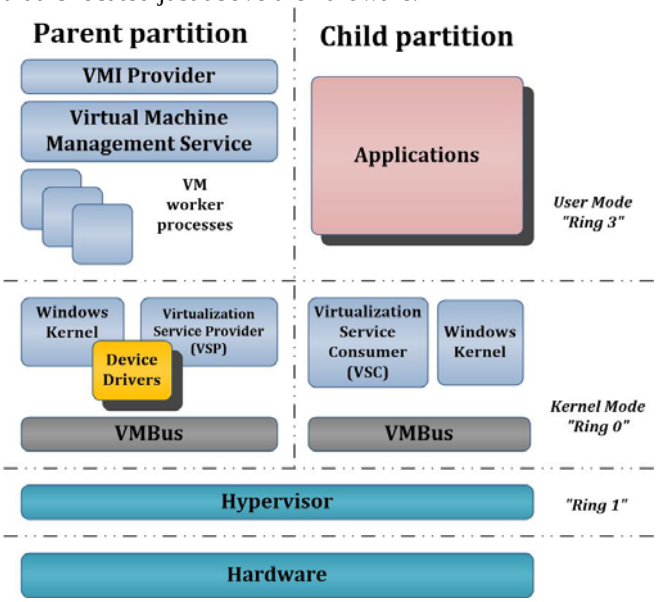

Figure 4. Hyper-V architecture

Just like a real kernel, Hyper-V manages the memory, the threads, and the basic system performance. Hyper-V supports the isolation of VMs and uses the partitions in which the OS will execute. There is a basic parental partition as real hOS and child partitions as gOSs. Hyper-V implement two kind of virtualizations, FHV (MS original) and PV (paravirtualization) for MS Windows OS which use VM bus, VSP and other PV components.

IV. Mathematical Modeling of FS Performance for Virtual Environment

The proposed filesystem performance mathematical modeling encompasses: (1) the WL (Workload) characteristics; (2) characteristics of the VMs (with the characteristics of the accompanying gOS and gOS FS); (3) modeling of the hypervisors; and (4) characteristics of the hOS and hOS FS (where the hypervisor operates as kernel in hOS).

The modeling is accomplished in three phases. We start from FS performance in the context of WL, then in the context of FS characteristics and finally in the context of the VE characteristics.

In order to evaluate the FS performance, it can be used the benchmark or real application testing approach, where all testing procedures generate specific WLs in FS.

For each workload, we used the parameter T_W as the total workload processing time. T_W is defined from the three point of views. The first point of view refers to kind of disk cycles in WL, calculated using the equation (1):

$$T_W = T_{RR} + T_{SR} + T_{RW} + T_{SW} \tag{1}$$

where, T_{RR} and T_{SR} denote the time components for random and sequential reading and T_{RW} and T_{SW} are the components for random and sequential writing. Writing can be synchronous or asynchronous, so writing performance depends significantly on the FS cache characteristics.

The second point of view refers to the analysis of the workload time, but in the FS context. It includes at least six components, as shown in the equation (2):

$$T_W = T_{DIR} + T_{Meta} + T_{FL} + T_{FB} + T_J + T_{HK} \tag{2}$$

where, T_W represents the total processing time to complete all operations for this workload. T_{DIR}, T_{Meta}, T_{FL}, T_{FB}, T_J, T_{HK} represent the processing time components required to perform all operations related to the directory, metadata, free lists, file blocks, journaling and house-keeping operations in the FS, respectively.

The third point of view refers to the analysis of the workload processing time when considering the VE with hypervisors influence. FS performance characteristics in VEs depend on a larger number of factors, as there is an impact of a range of virtualization effects that the applied virtualization technology enforces. Additionally, considering the context of the hypervisor VE environment, the overall data path becomes quite complex. Overall data path of the WL relies on five components: benchmark, gOS FS, hypervisor, VM image file and hOS FS.

The first component, the benchmark application, is running in VM and generates the workload, which can be assumed as a function of benchmark request characteristics, and gOS FS processing, which includes virtual disk drivers. The gOS output workload is further redirected to the hypervisor, which maps it to a large VM image file, and further redirects it to the hOS FS that generates requests

towards physical disk drivers. The whole data path depends on various factors, such as: characteristics of gOS-FS/hOS-FS, gOS file caching/hOS file caching (with specific cooperation of these two caches), hypervisor interconnection of physical and virtual disk drivers, hypervisor-CPU scheduling and other.

For *Tw* in VE, there are five components that have an impact to the workload time (equation 3):

$$T_w = f(bench, gOS-FS, VH-proc, Hyp-proc, hOS-FS) \quad (3)$$

1. Benchmark (*bench)* represents the interaction between the benchmark and gOS FS. The chosen benchmark will generate WL with the random and sequential components.

2. For the guest OS FS time component, *gOS-FS* and *hOS -FS* component, 2[d] and 5[th] component, the time for both OS-FS processing is represented by the function of FS processing and FS cache processing (equation 4):

$$g/hOS-FS = f(FSproc, FScache) \quad (4)$$

3. Virtual hardware processing, *VH-proc*, represents the processing time of the virtual disk hardware, and depends on the connection between the virtual disk drivers for gOS and the physical disk drivers of hOS.

VH-proc strongly depends on the type of virtualization. We are interested in two types of virtualization, FHV (full hardware virtualization) and PV (paravirtualization). FHV represents the full hardware emulation for storage components (storage hardware emulated in software). The guest disk drivers pass through virtual disk hardware, which contacts hypervisor for services in host OS. FHV shows great flexibility because gOSs are unchanged, but full hardware emulation in the software and a large number of context switches between gOS and hOS cause low FS performance.

PV requires the big changes in gOS as well as in host OS. PV drivers are created on the gOS and hOS side, fast asynchronous channels are established between them (IO rings for Xen, VM bus for Hyper-V). Fast channel and a small number of context switches between guest and host OS cause high FS performance, but the disadvantage of PV are the necessary major changes in OSs.

4. Hypervisor processing time, *Hyp-proc*, is the time necessary for the hypervisor to receive the requests from the VM (virtual disk driver) and forward them to the host physical disk drivers. FS requests from the guest FS (gOS-FS) are forwarded to the host FS (hOS-FS), via hypervisor and mapping through the VM image file. Many hypervisor parameters can affect FS performance.

5. Host OS FS processing, *hOS-FS*, is a component targeting the host FS processing. It works with big VM image file and is a function of FS processing and cache processing effects (equation 5):

$$hOS-FS = f(hOS-FSproc, hOS-FScache) \quad (5)$$

Actually, some components of equation (3) are closely interrelated, especially 2[nd] and 5[th]. As kernel for VE (Virtual Environment), each hypervisor has the hOS, which provides it with virtual disk drivers and physical disk drivers. The hOS can have one or several hOS FS types as a choice for the application.

Thus, in hypervisor-based VE there is always a FS pair to consider, *gOS-FS/hOS-FS*, therefore the 2[nd] and 5[th] components are in very complex interaction. These components rely on a pair of chosen FSs, whereas the number of the combinations in pair theoretically can be extremely large. Usually, there is a need to consider an interaction between two FS caches. The interaction of 2[nd] and 5[th] component as FS-pair is given in equation (6):

$$FS-pair = f(gFSt/hFSt, gFSc/hFSc, HypFSparm) \quad (6)$$

The first component in equation (6) comprises a FS pair, (*gFSt* is guest OS FS type, *hFSt* is host OS FS type) where each FS type in a pair has specific characteristics. There is a number of modern FSs that can be implemented as g/h-OS FSs. Most of them are 64-bit, extent based, and use accelerating techniques for allocation and searching (H-tree/B-trees). For write/update method, FSs use the overwriting or CoW (Copy on Write) techniques. The performance of the FSs highly depend on their own file caching characteristics, journaling methods, and different tunable parameters. FSs expose constant development, new versions are appearing, constantly.

For VE, there is need to emphasize the importance of the choice for the OSs, as there are Linux-like or MS Windows-based. Linux supports almost 100 types of FSs, including the most popular ext4, xfs, jfs, btrfs, wafl, zfs, and F2FS. MS Windows OSs implement only one NTFS, whereas rarely one can come across the FAT-32. The typical Linux-based hOS hypervisors are ESXi, Xen and KVM, whereas Hyper-V can be found as MS Windows-based. In this context, if we analyze FS pairs (gOS FS on hOS FS), the following can be assumed:

- for Linux hypervisors and Linux guests, there are a very large number of pairs (gOS FS on hOS FS)
- for MS Windows hypervisors and Linux guests, there are still large number of combinations for pairs (gOS FS on NTFS)
- for MS Windows hypervisors and Windows guests, there is only one pair (NTFS on NTFS)

The true is that there is no universal/best FS pair due to a large number of factors in the FS itself, and we expect that the optimal FS pair highly depends on WL characteristics.

The second component of equation (6) *gFSc/hFSc*, (gFSc is guest FS cache, hFSc is host FS cache) is a pair of two FS caches, precisely the interaction of these two caches. In hypervisor VE, we detect FS pair, and therefore two FS caches exist, FS cache on gOS and FS cache on hOS. These caches can be cooperative or exclusive. Two caches can be in the cooperation with WB (Write Back) or WT (Write Through) semantics, or excluded (none mode), when hypervisor excludes hOS FS cache. For exclude mod, hypervisor exclude host FS cache for VM data and thus frees RAM space for VMs. The hypervisor parameters determine the behavior of the two FS caches between each other.

The third component of equation (6) corresponds to the hypervisor (kernel) tunable parameters for FS in VE (HypFS

parm are Hypervisor FS parameters).

Each hypervisor has a number of kernel tunable parameters, and some of them can affect FS performance. Besides the two FS caches impacts, there is need to emphasize CPU scheduling, which is very important in the case of a large number of VM running. All hypervisors have their own schedulers, for example: Borrowed Virtual Time (BVT), Simple Earliest Deadline First (SEDF), Credit Scheduler are used by XEN, whereas CFS is used by KVM, as Linux Kernel's native CPU scheduler [23-25].

The storage space for VM image files has a lot of impact on FS performance. We consider DAS, a storage space created from local disks in a server, where the combination of SSD/HDD technology and RAID implementation has a great impact. However, VM images can be in NAS/SAN storage systems, which are based on network storage protocols such as NFS, SMB, FC, and iSCSI.

Also, the performances are affected by the HW extension for virtualization in the CPU itself and the chipset (Intel VT-x, VT-d), which depend on the model/version of the implemented CPU. With each new CPU model, a new version for HW extension for virtualization appears.

Finally, an impact is obvious when assuming the appearance of new versions of FSs, gOSs and their kernels, hypervisors and accompanying hOS, and CPU models with HW extensions that are being developed.

When all VE factors are considered, we cannot expect that there is a universal type-1 hypervisor, which is the best for FS performance. Hypervisors with PV are very promising for PV guests, such as PV Linux guests on Xen Hypervisor, or MS Windows guests on Hyper-V. In the case of FHV, the choice of the hypervisor strongly depends on WL characteristics. In the future, the advent of new versions of VE factors is also strongly influencing the choice of the optimal hypervisor.

We also expect that it is very complex to choose the optimal FS pair (gOS-FS on hOS-FS) in VE, whereas all depends mostly on WL characteristics.

V. Case Study and Expected Behavior

The discussed CS relies on the examination of four dominant type-1 hypervisors, ESXi, Xen, Hyper-V, and KVM, in full virtualization mode, and having the same Linux OS as gOS, with some default parameters for all hypervisors. The chosen benchmark testing software is Filebench.

All four hypervisors belong to the type-1 category, rely on the microkernel architecture, and are very thin and without drivers, except in the case of the KVM. The gOS FS type is ext4, whereas hOS FSs are ext4 or NTFS. There are two possible FS pair, {ext4 on ext4}, and {ext4 on NTFS}, referring to equation (6).

Next, we are discussing in further detail the five components that have an impact to the workload time in equation (3).

1. *bench*: All the candidates have the same WL. As the performance measurement relies on the use of the identical benchmark environment, VMs, gOS, gOS-FS (in this case the ext4), it is expected that this component has an identical effect on *Tw* for all the tested hypervisors.

2. *gOS-FS***:** time needed for gOS FS processing, *gOS-FS* is a function of the chosen FS characteristics, in this case ext4 (equation 7):

$$gOS - FS = f(ext4proc, ext4 - FScache) \tag{7}$$

Since the installed VMs for all the evaluated hypervisors are identical and based on ext4 as the gOS FS, this time value is almost the same for all tested hypervisors.

3. *VH-proc*: Depending on the used hypervisor, it relies on a specific FHV type: ESXi relies on the VMware FHV, KVM relies on the QEMU FHV, Xen relies on the QEMU FHV, Hyper-V relies on the Microsoft FHV.
Although all of these hypervisors rely on the FHV emulations, the performance will be different, because each hypervisor deploys FHV in its own way. KVM and Xen use open source QEMU FHV, whereas ESXi and Hyper-V implement the own solutions.

4. *Hyp-proc***:** Each hypervisor has its own processing delay, thus brings a certain small, but different overhead.

5. *hOS-FS***:** three out of four evaluated hypervisors (ESXi, KVM, Xen) rely on the use of the same type hOS-FS, in this study the ext4. Thus, it is expected that for ESXi, KVM and Xen this component will consume the similar processing time. On the other side, Hyper-V implements very different hOS FS, NTFS. This time is different for all the hypervisors, but shows the greatest differences in the case of Hyper-V. Each hypervisor also has its own FS caching in hOS.

Generally, all five components from the equation (3) will affect the performance of the tested hypervisors. Some of them have a similar impact, and some will cause solid performance differences. As all the tests are focused on the performance of natively virtualized guests (FHV emulation), for all of the evaluated hypervisors it is expected the dominant influence of the 3rd (*VH-proc*) and 4th (*Hyp-proc*) component from the equation (3), but also the 5th component is significant. In the case of Hyper V and NTFS FS it is expected the stronger influence of the 5th component (*hOS-FS*), as a difference from the other tested hypervisor.

Regardless of the fact that Linux based hypervisors have the same type of hOS FS (ext4), these are not the same versions and each hypervisor has their own host FS caching, so the 5th component will be different. In the context of equation (6) (1st and 2nd component), we expect Linux based hOS to operate faster with Linux gOS (ext4 on ext4) compared to Hyper-V (ext4 on NTFS).

In general, for our CS, we expect Linux based hypervisors to be better than Hyper-V. Among Linux based hypervisors, we expect there is no best/optimal hypervisor, but the performance depends on WL characteristics.

We expect all hypervisors to have a solid drop in FS performance compared to Native OS, due to FHV. The performance drop depends on the WL.

VI. Test Configuration and Benchmark Application

The assumption for the fair-play testing procedure is based on the use of the identical hardware, the OS on the guest or host side and the same benchmark measurement. The hardware configuration of the testing server and its components are shown in Table I. As guest OS we have chosen the CentOS 6.3 as a free Red Hat Enterprise Linux

(RHEL) version. Our hypervisors for the competition are as follows: ESXi 6.0, Windows Server 2016, Linux QEMU-KVM version 1.5.3, kernel 3.10.0-862 and XenServer 6.5.

TABLE I. SERVER REST ENVIRONMENT

HPE ProLiant DL 180 G6	
Component	**Characteristics**
Processor	Intel Xeon E5520 Quad Core 2.26GHz
Memory	24 GB DDR3
Cache	8MB L3
Hard Disk	Seagate Barracuda ES.2, 7.2k, 750GB SATA
Network	2 x 1Gb/s

All hypervisors were installed on the hard disks (DAS), where one part of DAS storage serves exclusively for servicing the hypervisor platform, and the other as a repository of data, applications and VM images. The two identical disk drives were mounted on the server HPE ProLiant DL 180G6. The first disk drive is used for hypervisor (XenServer, Hyper-V, KVM or ESXi), whereas the other is used as the storage repository for installed VMs. The same hardware was used for all hypervisors.

All hypervisors were installed on the hard disk with characteristics provided in Table II.

TABLE II. HARD DISK TEST ENVIRONMENT

Seagate Barracuda® ES.2	
Component	**Characteristics**
Model Number	ST3750330NS
Capacity	750 GB
Interface	SATA 3 Gbps
External tr. Rate	3 Gbps
Max Sustained	105 Mbps
Cache	32 MB
Avg. latency	4.16 msec
Spindle Speed	7200 rpm
Av. read seek time	8.5 msec
Av. write seek time	9.5 msec

The characteristics of the VM are shown in Table III. All VMs are identical.

TABLE III. VIRTUAL MACHINE PARAMETERS

Component	**Characteristics**
vCPU	4
Memory	6 GB
HDD	2 partitions – 180 GB
sda1	system partition with CentOS - 20 GB
sda2	additional test partition for testing - 150 GB
Swap	10 GB

Performance tests are carried out using the Filebench benchmark. It is a modern benchmark software designed to measure the FS performance of storage resources. Filebench is capable of generating multiple workload types in order to simulate different service environments such as web server, email, file server, database, and random file access.

We have chosen the four workloads for Filebench software: web, email, fileserver, and random file access (RFA) access. All workload environments rely on the use of a modified source files: File Server (fileserver.f), Web Server (webserver.f), Mail Server (varmail.f) and Random File Access - RFA (randomfileaccess.f). The fundamental parameters for chosen workload files, which provide realistic application conditions, are shown in Table IV. In order to achieve accurate results, for each workload test, the duration is set to 120 seconds.

TABLE IV. PARAMETERS OF THE SOURCE CODE *F FILE

	Fileserver	**Webserver**	**Varmail**	**RFA**
Nfiles	10000	1000	1000	10000
meandir width	20	20	1000000	20
meanfile size	128000	16000	16000	Random
Nthreads	50	100	16	5

VII. TEST RESULTS AND ANALYSIS

The goal of this CS is to measure the FS performance for popular virtualization platforms such as Xen, KVM, Hyper-V, and ESXi. Our interest is to evaluate the situation where more than one instance of VMs is used, when we can prove that the number of VM instances causes significant performance decrease, for all kind of hypervisors. All four virtual platforms have been tested with the guest in full hardware virtualization environments.

We have chosen the four workloads for Filebench software: web, email, fileserver, and random file access (RFA) access. Initially, we have tested all hypervisors with one running VM. The testing procedure is further repeated for the case of the 2 and 3 running VMs. With the same VMs we provide fair play conditions, as stated in equation (7). The test results of different workloads as well as of the native OS, are shown in Tables V - VIII and Figures 3 - 6.

TABLE V. WEB SERVER BENCHMARK RESULTS

Webserver [MB/s]			
Native OS	**71.43**		
	1VM	**2VM**	**3VM**
ESXi	57.1	51.7	40.5
KVM	14.7	12.9	11.6
Xen	35.9	35.1	31.3
Hyper-V	28.5	23.3	21.4

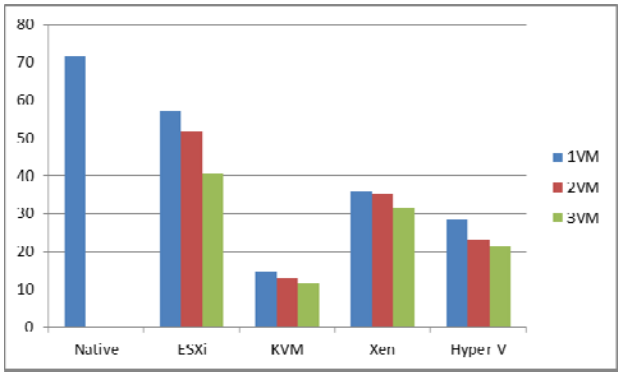

Figure 5. Native and four hypervisors Webserver test results

For the Webserver workload, which is characterized by a dominant random read component and a small amount of random write data (asynchronous writes) as covered in equations (1) and (2), the results clearly show that ESXi is totally superior when compared to other evaluated hypervisors. It is followed by Xen, then the Hyper-V, whereas KVM shows far the worst results.

TABLE VI. MAIL SERVER BENCHMARK RESULTS

Mailserver [MB/s]			
Native OS	**8.46**		
	1VM	**2VM**	**3VM**
ESXi	4.9	2.7	1.7
KVM	4.5	2.4	1.3
Xen	5.4	2.9	2.3
Hyper-V	1.4	0.5	0.3

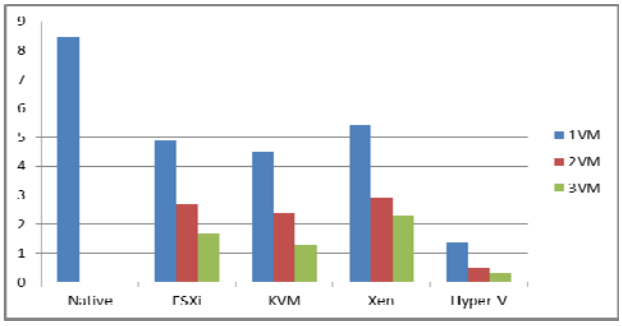

Figure 6. Native and four hypervisors Mail Server test results

For the Mail server workload, which is characterized by the dominant random reads and random writes, where random writes are represented by the synchronous transfers covered by equation (1) and (2), the results clearly show that Xen hypervisor provides the best performances, whereas it is followed by ESXi and KVM, whereas Hyper-V shows the worst results.

TABLE VII. FILE SERVER BENCHMARK RESULTS

Fileserver [MB/s]			
Native	**114.52**		
	1VM	**2VM**	**3VM**
ESXi	29.4	15	12.5
KVM	56.9	54.9	49.4
Xen	21.2	13.2	8.6
Hyper-V	28	13.8	8.8

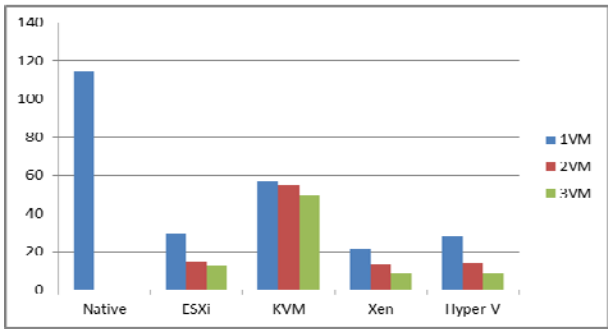

Figure 7. Native and four hypervisors File Server test results

For the Fileserver workload, which is characterized by all kinds of data transfers, when considering equations (1) and (2), the obtained results clearly show that KVM performs with far the best results. It is followed by ESXi, then with Hyper-V, whereas Xen hypervisor shows the worst results. KVM is remarkably the best, whereas the differences between the other three hypervisors are slight, especially in the case of the larger number of VMs.

TABLE VIII. RFA BENCHMARK RESULTS

RFA [KB/s]			
Native	**12067.74**		
	1VM	**2VM**	**3VM**
ESXi	6863.8	6312.5	5303.8
KVM	5767.4	5618.5	4833.1
Xen	1286.9	1125.8	1005.9
Hyper-V	3466.9	2417.1	1499.2

The Random File Access workload is characterized by the dominant random components (*RW* and *RR*), where random writes mostly represent the asynchronous transfers and rely on the FS cache, equations (1) and (2). The obtained results clearly show that ESXi hypervisor is convincingly the best option, whereas the system relying on KVM is also offering

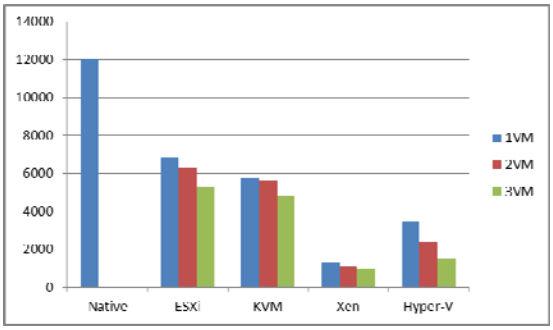

Figure 8. Native and four hypervisors File RFA test results

very good performances.

Hyper-V provides less acceptable performances, whereas Xen shows the worst results.

We will now present the result analysis for four hypervisors and four workloads.

First, let's analyze the performance drop caused by hypervisors relative to the native OS. For native OS, in equation (3), there are no 2nd, 3th and 4th components. Performances for native OS are a function of benchmark and host OS FS. Thus, native OS doesn't contain two important factors that dominate VE: large image file for VM and FS pair. If we look at the FS performance for native OS and the case for only 1VM when applying any of the selected hypervisors, it is noticeable a solid drop of FS performance in the case of the hypervisors. It is most pronounced for fileserver and RFA WLs, and slightly less for webserver and mailserver WLs. We believe that this is a consequence of the nature of WLs, due to the reduced benefit of FS cache. The webserver is dominated by *RR* cycles, and the mailserver is dominated by synchronous *RW* cycles. For both types of cycles, the beneficial impact of FS cache is very small.

Let's explain a connection between the mathematical model and Filebench workloads (WLs). Webserver workload is dominated by random read components, *RR*. Due to the *RR* dominance, the FS caches on both sides (guest/host) have no effect, thus the FS performance is determined primarily by the 3rd (*VH-proc*) and 4th (*Hyp-proc*) component of equation (3). The webserver workload FS performance is also under the influence of the 5th (*hOS FS*) component of equation (3) and the specific physical characteristics derived from the FS types in the interactive FS pair, reflected through the 1st component (*gFSt/hFSt*) of equation (6).

Mail Server workload is dominated by the synchronous *RW*, so there is no FS writeback cache effect. We consider that for mail server workload the FS performance is determined by the 3rd, 4th, 5th components of equation (3), and also by the characteristics of the interactive FS pair that are determined with the 1st component of equation (6).

The one of the most complex environments, File Server workload is dominated by all the components (*RR*, *RW*, *SR*, *SW*). FS workload is affected by most of the components (namely 3rd, 4th, and 5th) of equation (3) and the 1st and 2nd component of equation (6). Especially significant is the interactive influence of both FS caches defined by equations (4) - (6), as well as the mutual interactions of these two caches, defined in 2nd (*gFSc/hFSc*) and 3rd (*HypFS parm*) components of equation (6).

Random File Access workload is dominated by random read and asynchronous random write components. For asynchronous *RW* components it is typical the powerful influence of the writeback FS cache effect, which is defined by equations (4) - (6). RFA WL is also influenced by the interaction of two FS that make the pair (*gOS-FS/hOS-FS*), which is defined as 1st component of equation (6) and as 2nd and 5th components of equation (3). There is also noticeable mutual interaction of two FS caches, defined by 2nd and 3rd components of equation (6).

If we analyze the hypervisors individually in the context of the used workload, we can state the following:

1. ESXi is the best option for two analyzed workloads (Webserver and RFA workloads) and in two WL cases the second-best option (Mail server and Fileserver workloads), so ESXi could be nominated as the best hypervisor for our CS.

2. Xen was best for one workload (Mail server workload), but was also the worst option in the case of two other workloads (Fileserver and RFA workloads).

3. KVM was best option for only one workload (Fileserver workload), and the worst option for one workload (Webserver workload).

4. Hyper-V was mostly third in a row for the most optimum hypervisor, whereas for one workload it was the worst option (Mail Server).

As all the evaluated hypervisors are thin and microkernelized, it is expected to obtain the similar effects from the 4th component (*Hyp-proc*), whereas the strongest influence has the 3rd component (*VH-proc*), as in this case it relies on the full hardware virtualization in combination with two FS caches.

We expect that the 2nd and the 5th components in equation (3) and 1st and 2nd components in equation (6) provide similar effect on ESXi, Xen, and KVM since the used VMs and host FSs provide the similar characteristics for these hypervisors (same guest FS and similar host FS, (ext4 on ext4)). Hyper-V uses a completely different host FS, such as NTFS, so it is expected some different impact of the 5th component in equation (3), and FS pair (ext4 on NTFS) and cache pair in equation (6).

Thus, besides ESXi, all the other evaluated hypervisors have shown high sensitivity to the workload choice. For all the evaluated hypervisors and four types of workload, it is noticeable that the performance solidly decreases with the increase of the number of VMs (one, two and three VMs in our case). The effects of the file caching are strongly noticeable for all tested hypervisors.

General results overview for each hypervisor separately are following.

ESXi: The deeper analysis of the obtained results distinguishes ESXi as far the best of all hypervisors, for the workload which is dominated by random read component (Web Server) (Fig. 3), and workload that is dominated by random read and asynchronous random write components (RFA) (Fig. 6). In the case of the File Server and Mail Server workloads, ESXi performances are slightly weaker, making it the second-best choice. It is obvious that particularly the 3rd (*VH-proc*) and in some percent the 4th (*Hyp-proc*) components of the equation (3) have an impact on the random reads, whereas two interactive FS caches with writeback feature have the strongest impact to random writes. All these characteristics make ESXi the best option effect when compared to other evaluated hypervisors.

We think ESXi has an excellent own FHV, which is expressed in 3rd component (*VH-proc*) of equation (3) and therefore it wins solidly for the workload with *RR* (Web Server). We also think that ESXi makes the best use of the writeback FS cache effect compared to other hypervisors, so it wins in workload with dominant asynchronous *RW* components (RFA). The ESXi virtual disk drivers, when combined with writeback FS caching on the hOS side, can operate more efficiently for *RR* and asynchronous *RW* components when compared to the Xen/KVM QEMU and MS Hyper-V virtual disk drivers, equations (3) – (6). Thus, VMWare ESXi full virtualization with file caching is considerably more efficient when compared to QEMU full virtualization and Hyper-V full virtualization.

KVM: At the other side, KVM is far the best choice when there is need to operate in one of the most complex environments, such as the File Server workload, which is dominated by random and sequential components (Fig. 5). For such workload we have a combined effect of most components (3rd, 4th, 5th) in equation (3) and (1st and 2nd) in equation (6). So, in the combined impact of most of the VE components, where many cache benefits are expected, KVM has proven to be excellent. But for workloads where the FS cache effect is weak, the KVM FS performance are also weak. Thus, KVM is recommended to be avoided for the case of the workload with dominant *RR* and for those with dominant synchronous *RW* component, which means that KVM is poor choice for workloads with weak cache impact (Web Server, Mail Server).

Xen: Xen should be chosen in the case of the Mail Server environment, which is dominated by synchronous *RW*s (Fig. 4), meaning that all the writes to the disk are forwarded to the hOS FS (hOS-FS) disk driver. However, it is highly recommendable to avoid it when dealing with Random File Access workload, which is dominated by asynchronous *RW*s. Our experiment shows that Xen is very sensitive to the nature of the *RW* component. Xen is the best option when working with synchronous writes, as in that case when the system skips the file caching. On the other hand, Xen is the worst option for the case of working with asynchronous writes, as these intensively use the cache effects. Xen has an excellent impact of the 4th component (*Hyp-proc*) in equation (3) and a relatively good QEMU-Based FHV (*VH-proc*), so we think that it is the best candidate for Mail Server workload and second-best option for the Web Server workload. But, when many VE factors are involved, especially cache effects, Xen performances decrease. Definitely, Xen uses writeback cache effects significantly less than others, so it is the worst option for RFA and FS workloads.

Hyper-V: Unfortunately, neither of the evaluated workloads has proved the Hyper-V as the best choice, whereas it is highly recommendable to be especially avoided when working in workloads with synchronous *RW* (Mail Server) environments. For synchronous *RW* workload, without WB cache effects, we detected worst impact on Hyper-V mostly because of the 3rd, 4th, and 5th components of the equation (3), and 1st and 2nd components in equation

(6). We consider the main reason of this behavior is the hOS FS (NTFS) component in equation (3) and FS pair (ext4 on NTFS) component in equation (6).

Definitely, ESXi has a great *VH-proc* component and writeback cache effect, Xen has a great *Hyp-proc* component but is inferior with FS caching, KVM does well in the most complex workloads with the combined influence of many VE factors with strong FS cache effects, and Hyper-V is inferior to Linux guest due to its NTFS as hOS FS and FS pair (ext4 on NTFS).

VIII. Conclusion

Virtualization has brought great efficiency, flexibility, reliability with huge savings in power consumption. Therefore, virtualization has made significant changes in the computer industry and information technology (ICT). Certainly, virtualization has a significant impact on performance and we have tried in this paper to examine this impact in specific conditions, such as for different hypervisors from the FS performance point of view.

Based on the proposed model, our definitive conclusion is that for different applications in the context of FS performance there is no optimal (the best) type-1 hypervisor, but the choice of the best drastically depends on WL. The reason is a complex VE with a large number of input variables, which are in the complex interaction.

The optimal pairing of FS types on the host and guest side is particularly complex. Administrators of VE have a complex problem to determine the optimal VE for their applications of interest. We consider that a pool with 4 type-1 hypervisors should be created. Then, for each hypervisor with own hOS a pool of hOS FS with different types should be created, on which they can place VMs and migrate if necessary. For VMs with gOS, possible variation of gOS FS types should also be provided.

Administrators have two serious and intensive tasks. The first is to determine the optimal hypervisor for the applications of interest. The second is to determine the optimal FS pair for applications of interest. The model can be used for hypotheses about expected behavior, a good benchmark can give relatively good results for validation, but only real-app testing can give real validation.

In addition, administrators should tune the hypervisor/kernel tunable parameters and monitor changes that occur with the introduction of new versions of: CPUs, gOS, gOS FS, hypervisors with hOS, and hOS FS. And after performance analysis of its applications of interest, it is necessary to migrate of VMs to the optimal hypervisor and optimal FS pair.

For our specific CS, our results fit into the previous conclusions. Results show that in case of FHV, with Linux as a gOS, there is no possibility to choose a hypervisor of type 1 that would be universally the best choice, because the performance mostly depends on WL characteristics.

The obtained experimental results show that there are remarkable differences between hypervisors that rely on their own full emulation (ESXi and Hyper-V) and hypervisors that are based on the open source QEMU full emulation, such as Xen and KVM.

Our recommendation for CS with Linux based full virtualized guests is as follows:

- For typical web application environments, it is optimal to use ESXi whereas avoiding KVM.
- For typical email application environments, it is optimal to use Xen whereas avoiding Hyper-V.
- For complex application environments such as fileserver, it is optimal to use KVM.
- For typical random file access environments with lot asynchronous *RWs*, it is advisable to avoid Xen.
- Compared to native OS (without hypervisors), we detected significantly lower FS performance of all hypervisors, most pronounced for fileserver and RFA WL.

Future work can take place in several directions, and these are practically new CSs.

One direction is the problem of optimal pairing FS types in VE. For choice of FS type for the hOS, there is a large number of candidates (ext4, xfs, jfs, btrfs, NTFS). Precisely, Hyper-V has only one candidate for the role, it is the NTFS FS. The remaining three hypervisors have a large number of candidates of which the most dominant in use are: ext4, xfs, jfs, and btrfs FS. But, for Linux as a gOS, a similar FS variation can be made on the gOS side. Future work may include a variation of the FS types on the h/g OS side.

Second direction for future work is to evaluate the CSs based on FHV with MS Windows guests. The similar CS is to perform an identical experiment with MS Windows guests, where the four mentioned hypervisors would similarly compete in identical conditions for the effects of full hardware virtualization, using a suitable benchmark program, as HD Tune Pro or ATTO Benchmark. The MS Windows guests function exclusively as a fully virtualized guests, except in the case of the Hyper-V hypervisor, when they become PV guests.

Third direction for future work is PV, CSs based on PV. As the true power of Xen and Hyper-V is expected to be shown when working with the solutions that rely on the paravirtualization, our future work will include testing of the full and paravirtualization cases for these for hypervisors.

Two cases of analyzing the effects of paravirtualization are particularly interesting. The first is referred to Xen hypervisor who supports full hardware virtualization for all guest types, whereas for Xen-aware PV guests Xen supports the PV. The second case is referred to Hyper-V hypervisor, which supports full hardware virtualization for all guest types and paravirtualization for Hyper-V PV guests such as most MS Windows server OSs.

Additionally, our model is open for upgrades.

We give the idea for creation of KDB (Knowledge Data Base) related to the FS performance in VE. After realization a large number of CSs, KDB can be created. Each CS is specific and gives FS performance results, whereas our model will help to predict and interpret the results.

Each CS can represent the FS performance for a specific VE, which includes: benchmark/WLs, 4 dominant type-1 hypervisors in their current versions with following hOS, CPU version, specific gOS version, FS pairs (guest on host FS), hypervisor parameters tuning. Based on the measured FS performance depending on WL, the optimal hypervisors, FS pairs, and hypervisor parameters can be proposed for certain CS. Very bad combinations in terms of performance are also detected, in that CS. The KDB should include a

large number of CS and will constantly grow over time. With a large number of CSs, the KDB will contain optimal combinations {hypervisors, FS pairs, Hypervisor-Parameters} for a number of WLs, which can serve administrators to create VE for certain system case.

REFERENCES

[1] C. Jiang, Y. Wang, D. Ou, Y. Li, J. Zhang, J. Wan, B. Luo, W. Shi, "Energy efficiency comparison of hypervisors," Sustainable Computing: Informatics and Systems, vol. 22, pp. 311-321, 2019. doi:10.1016/j.suscom.2017.09.005

[2] Y. Jin, Y. Wen, Q. Chen, "Energy efficiency and server virtualization in data centres," IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), pp. 133-138, 2012. doi:10.1109/INFCOMW.2012.6193474

[3] S. Pawar, S. Singh, "Performance comparison of VMWare and Xen hypervisor on guest OS," IJICSE, vol.2, num. 3, pp. 56-60, 2015. ISSN: 2393-8528.

[4] A. Kumar, S. Shiwani, "Guest operating system based performance comparison of VMWare & Xen hypervisor," International Journal of Science, Engineering and Technology, vol. 2, no. 5, pp. 286-297, 2014. ISSN: 2348-4098.

[5] A. Bhatia, G. Bhattal, "A comparative study of various hypervisors performance," International Journal of Scientific and Engineering Research, 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] E. Correia, "Hypervisor based server virtualization," Encyclopedia of Information, Science and Technology, 3rd Edition, IGI Global, pp. 1182-1187, 2015. doi:10.4018/978-1-4666-5888-2.ch112

[8] H. Kazan, L. Perneel, M. Timmermann, "Benchmarking the performance of Microsoft Hyper-V server, VMWare ESXi and Xen hypervisors," J. of Emerging Trends in Computing and Information Sciences, vol. 4, no. 12, pp. 922-933, 2013. ISSN 2079-8407

[9] M. Polenov, V. Guzik, V. Lukyanov, "Hypervisors comparison and their performance," Computer Science On-line Conf., Springer, Cham, pp. 148-157, 2018. doi:10.1007/978-3-319-91186-1_16

[10] R. Y. Ameen, A. Y. Hamo, "Survey of server virtualization," (IJCSIS) International Journal of Computer Science and Information Security, vol. 11, no.3, 2013. arXiv preprint arXiv:1304.3557

[11] A. Varasteh, M. Goudarz, "Server consolidation techniques in virtualized data centers," IEEE System J., vol.2, no. 11, pp. 772-783, 2017. doi:10.1109/JSYST.2015.2458273

[12] P. Kedia, R. Nagpal, "Performance evaluation of virtual environment with respect to physical environment," International Journal of Computer Applications (0975 – 8887), vol. 89, num. 11, 2014. doi:10.5120/15676-4425

[13] P. Vijaya V. Reddy, L. Rajamani, "Evaluation of different hypervisors performance in the private cloud with SIGAR framework," (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 5, num. 2, 2014. doi:10.14569/IJACSA.2014.050210

[14] F. Bari, R. Boutaba, R. Estaves, L. Granville, M. Podlesny, "Data center network virtualization: A survey," IEEE Communications Surveys & Tutorials, vol. 2, no. 15, pp. 909-928, 2013. doi:10.1109/SURV.2012.090512.00043

[15] R. Marabito, J. Kjallman, M. Kamm, "Hypervisors vs. lightweight virtualization: a performance comparison," Cloud Engineering (IC2E), International Conference on IEEE, pp. 386-393, 2015. doi:10.1109/IC2E.2015.74

[16] J. Hwang, S. Zeng, F. Wu, and T. Wood, "A component-based performance comparison of four hypervisors," 13th IFIP/IEEE International Symposium on Integrated Network Management (IM) Technical Session, pp. 269-276, 2013. ISBN: 978-3-901882-50-0 978-1-4673-5229-1, 978-3-901882-51-7

[17] A. Elsayed, N. Abdelbaki, "Performance evaluation and comparison of the top market virtualization hypervisors," IEEE International Conference on Computer Engineering and Systems, pp. 45-50, 2013. doi:10.1109/ICCES.2013.6707169

[18] W. Graniszewski, A. Arciszewski, "Performance analysis of selected hypervisors (Virtual Machine Monitors-VMMs)," International Journal of Electronics and Telecommunications, vol. 62, num. 3, 231–236, 2016. doi:10.1515/eletel-2016-0031

[19] S. A. Algarni, M. R. Ikbal, R. Alroobaea, A. S. Ghiduk, F. Nadeem, "Performance evaluation of Xen, KVM, and Proxmox hypervisors," International Journal of Open Source Software and Processes, vol. 9, num. 2, 2018. doi:10.4018/IJOSSP.2018040103

[20] ***, VMware, "The CPU scheduler in VMware vSphere® 5.1 performance study," Technical White Paper, 2013.

[21] L. Cherkasova, D. Gupta, A. Vahdat, "Comparison of the three CPU schedulers in Xen," ACM SIGMETRICS Performance Evaluation Review 35(2), pp. 42-51, 2007. doi:10.1145/1330555.1330556

[22] 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

[23] B. Djordjevic, N. Macek, V. Timcenko, "Performance issues in cloud computing: KVM hypervisor's cache modes evaluation," vol.12, no.4, pp. 147-165, 2015. doi:10.12700/APH.12.4.2015.4.9

[24] K. T. Raghavendra, "Virtual CPU scheduling techniques for kernel based virtual machine (KVM)," IBM India, IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 1-6, 2013. doi:10.1109/CCEM.2013.6684443

[25] L. Carvalho, "Windows server 2012 Hyper-V cookbook," Packt Publishing Ltd, 2012.

[26] V. K. Manik, D. Arora, "Performance comparison of commercial VMM: ESXi, XEN, HYPER-V & KVM," 3rd International Conference on Computing for Sustainable Global Development, 2016. ISBN: Electronic ISBN:978-9-3805-4421-2, DVD ISBN:978-9-3805-4420-5, Print on Demand(PoD) ISBN:978-1-4673-9417-8