Chapter 12
Performance Evaluation and Comparison of Hypervisors in a Multi-Cloud Environment

Nalin Reddy, R. K. Nadesh, R. Srinivasa Perumal, Nikhil Chakravarthy Mallela, and K. Arivuselvan

12.1 Introduction

Many cloud service providers and the arising business need have created an enormous demand for computing power, platform, and applications across the globe. Digital Business Technology (DBT) is a combination of Operation Technology (OT) and Information Technology (IT). Everything as a Service (XAAS) can be easily provisioned with the help of a virtual machine. All the information technology infrastructure needs can be seamlessly provisioned as a virtual environment to the remote users with the support of virtualization. Cloud computing offers software, platform, database, infrastructure, security, and anything in IT as a service. For all kinds of offerings in public, private, and hybrid cloud, virtualization is the basic building block for the multi-cloud environment. A hypervisor is a tool used to create an abstraction of the resources to provide a multi-tenant computing model in a multi-cloud environment.

Virtualization is a broad term that refers to an abstraction of resources across many aspects of computing. One physical machine to support multiple virtual machines that run in parallel is the purpose. Many enterprises prefer to have a virtualized environment because of the drawbacks in the traditional computing environment like too many servers for too little work, aging of hardware and end of usable life, high infrastructure requirements, and the limited flexibility in the shared environments. The core technology behind virtualization is the hypervisor, sometimes referred to as the Virtual Machine Monitor. Hypervisor was introduced in the 1970s

N. Reddy · R. K. Nadesh (✉) · R. Srinivasa Perumal · N. C. Mallela · K. Arivuselvan
School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India
e-mail: rnalin.reddy2017@vitstudent.ac.in; r.srinivasaperumal@vit.ac.in; karivuselvan@vit.ac.in

R. Nagarajan et al. (eds.), *Operationalizing Multi-Cloud Environments*, EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-74402-1_12

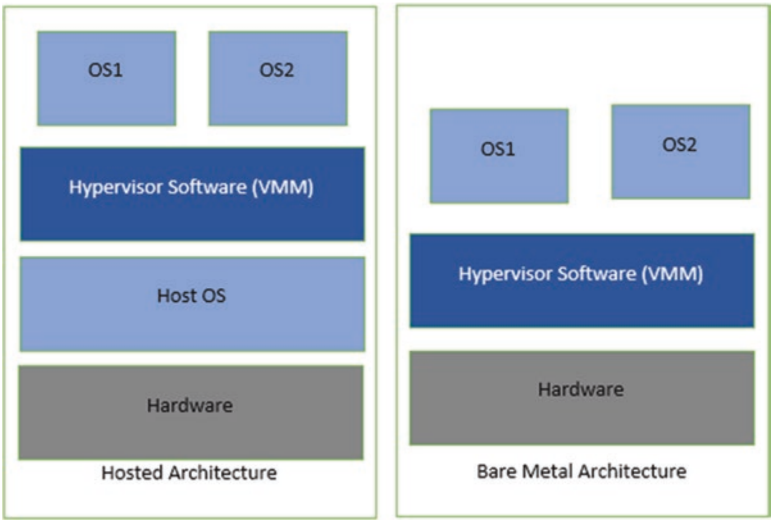

Fig. 12.1 Types of hypervisor

as part of the IBM S/360. A hypervisor is a computing layer that allows multiple operating systems to run a host computer simultaneously. The hypervisors are broadly classified as Type-1 and Type-2 hypervisor. The first type runs directly on the underlying host system, known as a bare-metal hypervisor or native hypervisor. The second of this kind has a host operating system running over the underlying host system and is also known as a hosted hypervisor (Fig. 12.1).

In the bare-metal hypervisor, the abstraction is done at the hardware layer and managed by the Virtual Machine Monitor (VMM). In the hosted system, the virtual machine is created, and the total control of the guest operating system is within the host operating system. This facilitates the option multi-tenant model with different operating systems and applications from multiple physical servers for the multi-cloud environment. Any user of private, public, hybrid cloud models can also share the resources among themselves in several regions where the computing model is called Inter-Cloud Computing. Server virtualization technology helps the enterprise provide platform, software, infrastructure, and security as a service in their private cloud owned by them or managed by the third-party service provider [1]. It is a promising technology to reduce the cost through server consolidation. Microsoft Hyper-V, VMware ESX are server-centric tools that tend to focus on the virtualization of the infrastructure. The other famous virtualization tools from VMware, Citrix, Oracle, and Amazon also play a significant role in hypervising the hardware. Each hypervisor has its features and limits, and choosing the appropriate hypervisors for digital business technology today is always challenging. Different virtualization types include server virtualization, storage virtualization, network virtualization, memory virtualization, and data virtualization in a multi-cloud computing environment. The multi-tenant deployed modes in data centers and application servers help the end user use the services from a fully isolated server, virtualized server, or shared virtualized server. The tenants can also share the database, schema, and server [2].

Virtualization can be done at any level, starting from the instruction set architecture level, hardware abstraction level, operating system level, user level API, and application level. Therefore, estimating CPU performance, memory, and disk helps the enterprise to choose the appropriate virtualization model by analyzing the number of iterations, number of threads, number of floating-point operations per second, and number of input/output operations per second. The memory performance is measured based upon the number of iteration, number of threads, throughput, size, and latency. The disk performance is calculated based on the read and write operations, size of the file, and time taken for read and write operations [3].

12.2 Related Works

Cloud computing continues to dominate by making the data and computing facilities available in an unprecedented economy, thereby increasing the end-user reliability and scalability. Nimbus and Open Nebula provide full freedom to build and manage enterprise cloud with advanced features for multi-tenancy, resource provision, and elasticity with virtualized services and applications. Open Nebula can integrate any hypervisor for any workload and server deployments. Xen and KVM are the two virtual managers taken for the performance measures, and results obtained prove that Xen outperforms in most cases than the KVM [4]. There are multiple challenges in configuring the hypervisor parameters to optimize its performance in the data center. The normal practice of tuning the scheduler parameter will not be effective in the highly dynamic workloads and utilization of the host machine resource. Discover, optimize, and observe were the three steps used to improve efficiency and termed as dynamic scheduler reconfigurations [5]. Cloud data centers have a collection of servers available to provide services to users at different times virtually. Continuous and uninterrupted services are essential, and therefore scheduling and maintenance activities in the data center are more difficult when the services are offered. A study was conducted to understand the server resources management, queuing the server requests with proper maintenance schedules for the server using an Open Nebula tool kit. Scheduling and maintenance algorithms are devised and deployed to estimate the system performance and evaluated through the Sunstone GUI server [6]. An experimental study was conducted with virtual machines, operating systems, and hardware and analyzed how configurations influence a deterministic algorithm [7]. The targeted system with multi-processor virtual machines placed on a commodity chip-multiprocessor and the performance degradation in a virtualized environment is analyzed considering the hypervisor slowdowns and the nearby VM neighborhood hypervisor noise. The aim is to reduce the noise level using different hardware and software techniques and ensure that they are ready for the cloud [8]. The use of virtual PC machine is considered in the context of interoperability with Network Controlled System, and specific application requests answer the need for virtualization, services, and security. Xen hypervisor is used to demonstrate the performance of the guest operating system and their control in the virtual environment [9].The online gaming system in the public cloud platforms was implemented as a virtualized cloud gaming platform supported with the latest software and hardware, and extended support is provided for both remote servers and local clients. This has achieved full-fledged deployment of gaming services, optimization, and integration of all the modules [10]. The workload-aware credit scheduling method is used for improving the network I/O performance in the virtual environment that gains better bandwidth and less response time, and fairness is achieved between I/O-intensive and CPU-intensive domains [11]. The distributed performance management schemes in virtual environments using non-linear controls have improved scalability among 10 virtual machines [12]. Variations in performance and scalability in IaaS clouds using multi-tier workloads in three hypervisors, namely, CVM, XEN, and KVM, show that each one has its own individual characteristics and performance according to their configurations [13]. Elastic applications for mobile cloud services with VM were implemented with a suspend and run the model. The front end screen reads the input and sends the same to back end server for processing. The processed result is displayed back on the user screen. This gives the user the feeling that the computation has happened in the mobile phone but was originally computed at the back end server [14]. Throughput based resource allocation model addresses the need for clustering and scheduling in a virtual environment. Virtual machine performance is computed, and based on the need, it is migrated to the next physical server available, thereby optimizing the server performance. This increases the overall efficiency of the virtual environment and has the key characteristics of cloud computing [15].The sharing of resources in a mobile environment with the help of remote method invocation is an added entity for the peer-to-peer computing model, which also supports the multi-cloud under heavy loads [16]. Feedback control for hypervisor-based multi-core is implemented with two controls. One control will limit the sharing of the resources, and the second control maintains the performance. These controls guarantee the execution of critical partitions, and it is recommended for the use of multi-core execution platforms [17]. Software-Defined Networking (SDN) plays a major role in separating the data plane from the control plane. Network Function Virtualization (NFV) allows multiple tenants the flexibility of sharing the physical resources. SDN and NFV are complementary technologies as virtualization allows the SDN networks to leverage the combined benefits of SDN networking and network virtualization. Therefore, centralized and distributed hypervisors are part of the SDN hypervisor [18]. Fault tolerance hypervisor is introduced to have a back-up hypervisor and the primary hypervisor running parallelly, and when the primary VM fails, the back-up hypervisor will start functioning within a time of 10 ms which is emerging approach for time-sensitive applications. Synchronization of both the virtual machines is done to provide economic and fault tolerant solution [19]. A virtual machine with a webserver running is used to study the performance degradation when Denial of Service (DoS) attacks happen in the network and it is observed that the performance degrades up to 23% when compared to non-virtual system. Generally, the performance degradation due to security threat is high when compared to the normal physical resources [20]. Project managers maintaining the infrastructure availability prefer to move virtualization to cloud computing to reduce capital expenses, operational expenses, and the total cost for ownership [21]. Common vulnerabilities, security, and threats during VM image sharing, VM migration, and secure host os were studied and recommended that the hypervisor should analyze the need for VM and ensure that unnecessary operations are not carried out [22]. Browser is the main entity of cloud computing environment where any user can directly connect themselves to the virtual environment. Minimizing the system resources while using the cloud-based application can be supported with browser add-on which is also secured and provides user friendliness [23].

12.3 Methodology

The virtual boxes VMware Workstation, Oracle VirtualBox, and Windows Subsystem for Linux (WSL) are used to study the virtual machine performance. VMware Workstation is a hosted hypervisor that supports the users to set up virtual machines on a single physical machine and run simultaneously. The hosted hypervisor runs on x64 version and x86–32 version of Microsoft Windows, Linux, BSD, and MS-DOS. The VMware Workstation has the free-of-charge version known as VMware Workstation player, and it can be used for non-commercial. However, the operating system needed to use is proprietary like Windows. VMware Workstation helps bridging the existing host network adaptors and sharing of host resources like physical disk drives, USB devices, and mounting of disk drives. The main benefit of using VMware Workstation is snapshot; the current state of the virtual machines can be saved and later resumed. One more added advantage of using VMware Workstation is grouping multiple images in an inventory folder; therefore the machines in the folder can be powered on and powered off as a single object and useful for client-server environments. The cross-platform virtualization software Oracle VM VirtualBox allows the system to run multiple operating systems at the same time. The latest release of Oracle VM VirtualBox 6.0 provides integration with the open source and the cloud development. Therefore, creating and deploying virtual machines can be done everywhere with upload and download option, review, and make changes offline. It also provides the developers to create multiplatform environments and also to develop applications for container within the Oracle VM VirtualBox on a single machine. The virtual machine can further be deployed to Oracle VM server in the case of server virtualization environments. Windows subsystem for Linux is compatible for running Linux binary executable natively on Windows 10 and Windows Server 2019. It uses the Linux kernel through a subset of Hyper-V features and allows the user to run Linux command-line tools alongside with the Windows command-line, desktop, and applications and access the Windows files from Linux. Each of the virtual machines is configured in three different players with certain workloads. For the execution of the task, each virtual machine is designed with one virtual CPU and 1GB of memory. The Windows Server framework was introduced inside the virtual machine. The 64-piece machine is favored since most of the hypervisors are perfect with the x64. HP Pavilion pc designed with an Intel i5-6200 U 2.40GHz processor with 8GB RAM is used for the study. Although the test framework has 8GB of RAM introduced, it is booted with just 2GB of RAM for local tests. Windows Subsystem for Linux is a compatibility layer for running Linux binary executables natively on Windows 10 and Windows Server 2019. The WSL has Full Linux kernel, increased file I/O performance, and full system call compatibility.

12.3.1 Setting Up of the Virtual Machines

1. Set up the initial system with Intel i7 processor and 8GB Ram.
2. Create a Virtual Machine over the host operating system.
3. Create three VM using VMware Workstation, Oracle VirtualBox, and WSL.
4. Kali Linux Operating System installed in all three Virtual Machines.
5. Key VM Configurations.

   Processors: 1VCPUs
   Memory: 2048 MB
   Network Adapter: Internal Virtual Switch
   Percentage of Total System Resources: 25%
   Clustered: No

6. Benchmark Tests, run on each Kali Virtual Machine to test the efficiency of the underlying Hypervisor.

12.3.2 Setting Up of the Benchmark Tests

Benchmark test is introduced to test the effectiveness and efficiency of the asset usage for the virtual environment.

The three benchmark tests are:

- CPU Benchmark
- Memory Benchmark
- Disk Performance Benchmark

12.3.2.1 CPU Test

A CPU benchmark (CPU benchmarking) is a progression of tests intended to quantify the presence of a PC or gadget CPU (or SoC). Many gauges or standard estimations are utilized to look at the exhibition of various frameworks, using similar strategies and conditions. A typical CPU benchmark test will test the framework against the gauges for the sort of CPU utilized. CPU particulars regularly estimated by a benchmark test incorporate the clock speed, the quantity of directions executed, library calls per cycle, and large engineering effectiveness factors. The benchmark norms change between ages of CPU and among Intel and AMD CPU. CPU benchmark programming will likewise assemble and give data on many principle gadgets in a PC framework, such as the processor, motherboard and chipset, and memory.

12.3.2.2 Memory Utilization Test

When a PC program needs to utilize an area of memory to store information, it makes a solicitation to windows for the measure of memory it requires. Windows designates the memory to the program (except if framework assets are deficient) and comes back to the mentioning program, the allotted square's primary memory opening. It is conceivable that a few projects may demand a lot of memory. The "Memory Speed Per Block Size" test, like the "Memory Speed Per Access Step Size" test, is made out of numerous means. During each test's progression, the performance test demands a square of memory and goes through the square estimating the entrance speed. memory benchmarking is a straightforward memory benchmark program, which attempts to quantify the pinnacle transfer speed of successive memory gets and the idleness of irregular memory brings. Data transmission is estimated by running diverse get-together codes for the adjusted memory squares and endeavoring distinctive prefetch techniques.

12.3.2.3 Disk Utilization Test

When PC clients get another hard drive plate, it is mandatory to know the hard drive's particular execution in their PCs. A plate execution test is done to understand the specific performance. Circle benchmarking is the way toward running programs that precisely measure move speeds under different plates in different situations. The point is to deliver figures in Mbps that abridge the speed attributes of a circle. There are a few free programming choices accessible, which you can, without much of a stretch, download and run yourself to benchmark your own drive.

12.4 Implementation

The proposed method is executed to study the CPU performance, memory performance, and disk performance. The virtual boxes VMware Workstation, Oracle VirtualBox, and WSL are used to review the virtual machines' performance. Each of the virtual machines is configured in three different players with certain workloads. All the three virtual machines created are executed to study the CPU performance w.r.t number of iterations, the number of threads, the number of Floating-Point operations per second, and the number of input/output operations second. The memory performance is measured based upon the number of iteration, number of threads, throughput, size, and latency. The disk performance is based on read and write operations, size of the file, and time taken for read and write operations. Kali Linux operating system was uniquely used in all the virtual machine, and the results obtained at the end of each iteration are tabulated.

12.4.1 Running Kali Linux on Windows Subsystem on Linux (Hypervisor—I)

The CPU benchmark testing is done using the code written in C/C++, and the obtained results are tabulated according to the iterations, number of threads, number of floating-point operations per second, and the number of I/O operations per second. A total number of 10 iterations are carried out to study the CPU performance for the hypervisor using Kali Linux on Windows Subsystem on Linux.

The memory performance is tabulated based on the memory benchmarking test parameters like size in bytes, number of threads, and throughput for the read and write operations for both random and sequential access for the number of iterations. The latency is also computed to understand the delay in read and write operations for the given task with the CPU performance for the hypervisor using Kali Linux on Windows Subsystem on Linux (Tables 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 12.8, and 12.9).

The performance of the disk is carried out to estimate the reading and writing rate in Mbps; the size of the file is also considered for the total read and write operations per milliseconds. Disk benchmarking test is conducted along with the CPU and memory test for the hypervisor using Kali Linux on Windows Subsystem on Linux.

Table 12.1 CPU performance

Execution count	Number of iterations	Number of threads (multithreading)	Number of floating-point operations per second (in G-FLOPS)	Number of input/output operations per second (in G-FLOPS)
1.	1,000,000	1	13.761542	13.866962
2.	1,000,000	1	13.924173	13.830580
3.	1,000,000	1	13.414729	12.084710
4.	1,000,000	1	13.668033	13.620210
5.	1,000,000	1	13.266403	13.605371
6.	1,000,000	1	13.361323	13.551228
7.	1,000,000	1	13.550177	13.753762
8.	1,000,000	1	13.720102	13.442020
9.	1,000,000	1	13.578925	13.306751
10.	1,000,000	1	13.662736	13.220409
SUM	10,000,000	10	135.908143	134.282003

12.4.2 Running Kali Linux on VMware Workstation (Hypervisor-II)

The CPU benchmark testing is done using the code written in C/C++, and the obtained results are tabulated according to the iterations, number of threads, number of floating-point operations per second, and the number of I/O operations per second. A total number of 10 iterations are carried out to study the CPU performance for the hypervisor using Kali Linux on VMware Workstation.

The memory performance is tabulated based the memory benchmarking test parameters like Size in bytes, Number of Threads, Throughput for the read and write operations for both random and sequential access for the number of iterations. The latency is also computed to understand the delay in read and write operations for the given task with the CPU Performance for the hypervisor using Kali Linux on VMware Workstation.

The performance of the disk is carried out to estimate the reading and writing rate in Mbps, the size of the file is also considered for the total read and write operations per milliseconds. Disk benchmarking test is conducted along with the CPU and memory test for the hypervisor using Kali Linux on Windows Subsystem on VMware Workstation.

Table 12.2 Memory performance

Execution count	Number of iterations	Number of threads	Size (bytes)	Throughput for seqread()	Throughput for seqwrite()	Throughput for randread()	Throughput for randwrite()	Latency
1.	1,000,000	10	10,240	1093.576680	1163.855930	1249.048243	1028.942180	0.923900
2.	1,000,000	10	10,240	1114.380148	1585.388570	1235.726828	1145.171190	0.890000
3.	1,000,000	10	10,240	1140.562354	1309.574123	1115.328405	1301.770329	0.910900
4.	1,000,000	10	10,240	1110.838498	1203.392437	1358.699061	1236.455398	1.246600
5.	1,000,000	10	10,240	1062.710044	1183.694756	1241.359062	1333.302816	0.987500
6.	1,000,000	10	10,240	1079.948504	1375.542437	1124.116638	1191.022263	0.928900
7.	1,000,000	10	10,240	1212.436839	1230.578571	1204.567490	1242.535845	1.407700
8.	1,000,000	10	10,240	1113.078924	1276.804871	1321.040630	1344.414386	0.975300
9.	1,000,000	10	10,240	1165.537709	1209.499971	1501.075084	1286.280667	0.952300
10.	1,000,000	10	10,240	1122.852707	1290.952293	1242.830390	1289.682061	0.952800
SUM:	10,000,000	100	102,400	11215.922407	12829.283959	12593.791831	12399.577135	10.1759

Table 12.3 Disk performance

Execution count	Size of the file (bytes)	Total write time (ms)	Total read time (ms)	Writing rate (Mbps)	Reading rate (Mbps)
1.	10	77.6674	101.277	0.128759	0.0987387
2.	10	67.5392	101.461	0.148062	0.0985596
3.	10	78.4833	97.7968	0.127416	0.102253
4.	10	75.2111	104.913	0.132959	0.095317
5.	10	80.4484	98.7593	0.124303	0.101256
6.	10	88.9543	114.768	0.112417	0.0871322
7.	10	83.6196	104.838	0.119589	0.0953855
8.	10	89.4208	96.3829	0.111831	0.10386
9.	10	60.0919	101.707	0.166412	0.0983218
10.	10	64.8395	102.022	0.154227	0.0980185
SUM:	100	766.2755	1023.925	1.325975	0.9788423

Table 12.4 CPU performance

Execution count	Number of ıterations	Number of threads (multithreading)	Number of floating-point operations per second (in G-FLOPS)	Number of ınput/output operations per second (in G-FLOPS)
1.	1,000,000	1	13.094113	13.547008
2.	1,000,000	1	13.573411	13.398396
3.	1,000,000	1	13.545767	13.622528
4.	1,000,000	1	13.451017	13.493908
5.	1,000,000	1	13.544296	13.556375
6.	1,000,000	1	13.147545	13.008343
7.	1,000,000	1	13.516356	13.464139
8.	1,000,000	1	13.140898	13.388689
9.	1,000,000	1	13.547628	13.606009
10.	1,000,000	1	13.179020	13.325062
SUM:	10,000,000	10	133.740051	134.410457

12.4.3 Running Kali Linux on Oracle Virtual Box (Hypervisor-III)

The CPU benchmark testing is done using the code written in C/C++, and the obtained results are tabulated according to the iterations, number of threads, number of floating-point operations per second, and the number of I/O operations per second. A total number of 10 iterations are carried out to study the CPU performance for the hypervisor using Kali Linux on Oracle VirtualBox.

The memory performance is tabulated based on the memory benchmarking test parameters like size in bytes, number of threads, and throughput for the read and write operations for both random and sequential access for the number of iterations. The latency is also computed to understand the delay in read and write operations

Table 12.5 Memory performance

Execution count	Number of iterations	Number of threads	Size (bytes)	Throughput for eqread()	Throughput for seqwrite()	Throughput for randread()	Throughput for randwrite()	Latency
1.	1,000,000	10	10,240	936.186777	964.873246	1048.628431	1005.201553	1.0519
2.	1,000,000	10	10,240	1025.652663	1140.252284	1103.009520	1065.409470	1.1450
3.	1,000,000	10	10,240	990.484107	1082.289312	842.771259	1136.113549	1.1146
4.	1,000,000	10	10,240	694.835332	1180.363595	1096.951564	1018.133799	1.1420
5.	1,000,000	10	10,240	864.021094	1023.750061	1128.411084	1075.297134	1.2156
6.	1,000,000	10	10,240	873.340274	1145.984699	1017.738523	1152.091414	1.1927
7.	1,000,000	10	10,240	778.944397	1018.331553	863.558575	1103.531888	1.2161
8.	1,000,000	10	10,240	1262.280005	1073.151162	974.467729	1151.332418	1.2967
9.	1,000,000	10	10,240	805.914995	960.278401	1138.457196	1109.369446	1.3303
10.	1,000,000	10	10,240	835.552014	1078.504500	950.399710	1167.679287	1.2021
SUM:	10,000,000	100	102,400	9067.211658	10667.778813	10164.393591	10984.159958	11.907

Table 12.6 Disk performance

Execution count	Size of the file (bytes)	Total write time (ms)	Total read time (ms)	Writing rate (Mbps)	Reading rate (Mbps)
1.	10	18.9476	3.8338	0.527772	2.60838
2.	10	22.049	3.89774	0.453534	2.56559
3.	10	22.8735	3.84719	0.437188	2.5993
4.	10	21.3802	3.81358	0.467722	2.62221
5.	10	23.6215	3.67262	0.423343	2.72285
6.	10	24.0222	4.06257	0.416281	2.4615
7.	10	23.3403	4.85422	0.428443	2.06006
8.	10	19.2859	3.75868	0.518514	2.6605
9.	10	18.1091	3.79127	0.552209	2.63764
10.	10	18.2845	3.81848	0.54691	2.61884
SUM:	100	211.9138	39.35015	4.771916	25.55687

Table 12.7 CPU performance

Execution count	Number of iterations	Number of threads (multithreading)	Number of floating-point operations per second (in G-FLOPS)	Number of ınput/output operations per second (in G-FLOPS)
1.	100,000,000	1	13.290265	13.336000
2.	100,000,000	1	13.392698	13.228865
3.	100,000,000	1	13.415576	13.240553
4.	100,000,000	1	12.925370	12.646173
5.	100,000,000	1	12.357365	13.339931
6.	100,000,000	1	13.127266	13.099025
7.	100,000,000	1	13.050231	13.145842
8.	100,000,000	1	13.267999	13.165307
9.	100,000,000	1	13.429680	13.074873
10.	100,000,000	1	13.383794	13.627305
SUM:	1,000,000,000	10	131.640244	13.903874

for the given task with the CPU performance for the hypervisor using Kali Linux on Oracle VirtualBox.

The performance of the disk is carried out to estimate the reading and writing rate in Mbps; the size of the file is also considered for the total read and write operations per milliseconds. Disk benchmarking test is conducted along with the CPU and memory test for the hypervisor using Kali Linux on Windows Subsystem on Oracle VirtualBox.

Table 12.8 Memory performance

Execution count	Number of iterations	Number of threads	Size (bytes)	Throughput for seqread()	Throughput for seqwrite()	Throughput for randread()	Throughput for randwrite()	Latency
1.	1,000,000	10	10,240	914.229914	1263.572935	1186.507496	1365.777923	1.1557
2.	1,000,000	10	10,240	1386.180184	1330.511356	1235.799646	1357.204245	1.2348
3.	1,000,000	10	10,240	1385.447579	1323.040818	1374.010352	1589.233101	1.2516
4.	1,000,000	10	10,240	1159.544399	1187.649790	1195.571518	1296.378809	1.2280
5.	1,000,000	10	10,240	1064.490127	1191.902245	1077.341005	1129.687567	1.2341
6.	1,000,000	10	10,240	1378.526260	1377.801721	1401.745873	1360.903310	1.1872
7.	1,000,000	10	10,240	1376.264602	1340.890026	1312.689034	1549.886926	1.2520
8.	1,000,000	10	10,240	510.516809	1653.905363	1381.614072	1522.986202	1.1990
9.	1,000,000	10	10,240	1224.614307	1550.345235	1440.945445	1514.189170	1.4194
10.	1,000,000	10	10,240	1134.331458	1498.822184	1263.953713	1391.976636	1.3616
SUM:	10,000,000	100	102,400	11534.145639	13718.441673	12870.178154	14078.223889	12.5234

Table 12.9 Disk performance

Execution count	Size of the file	Total write time (ms)	Total read time (ms)	Writing rate (Mbps)	Reading rate (Mbps)
1.	10,240	19.4234	5.66896	0.514844	1.76399
2.	10,240	8.74747	5.43169	1.14319	1.84105
3.	10,240	23.3246	4.49415	0.428732	2.22511
4.	10,240	21.6748	5.04888	0.461366	1.98064
5.	10,240	18.7458	6.4755	0.533453	1.54428
6.	10,240	18.7675	8.55579	0.532836	1.1688
7.	10,240	14.3616	6.74879	0.696303	1.48175
8.	10,240	20.0254	4.70196	0.499366	2.12677
9.	10,240	11.0351	6.95651	0.906202	1.4375
10.	10,240	17.0936	7.97497	0.585015	1.25392
SUM:	102,400	173.19927	62.0572	6.301307	16.82381

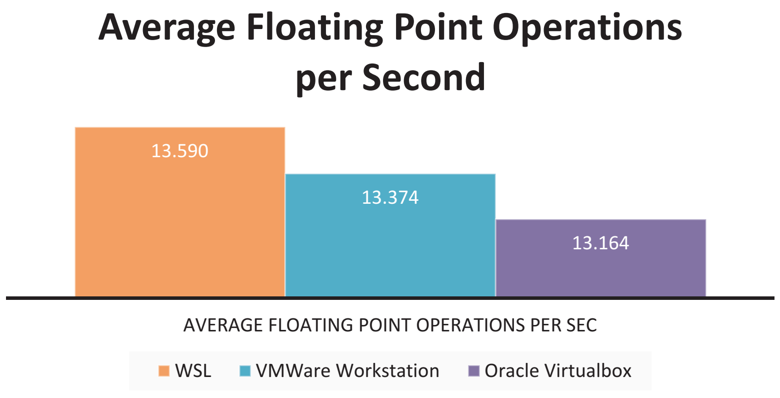

Fig. 12.2 Average floating-point operations per second – CPU performance

12.5 Results and Discussion

The results obtained after the execution of all the three virtual machines are summarized, and the average of each and every parameter is computed. The CPU performance of the three virtual machines is calculated based on the average floating-point operations per second. The averages of WSL, VMware Workstation, and Oracle VirtualBox are computed, and the average input/output operation per second (Fig. 12.2).

It is observed that the average flops obtained are less in Oracle VirtualBox when compared to the VMware Workstation and the WSL. However, the deviation is very less when compared. The I/O operations per second obtained and on computing the average of the input/output operations per second have yielded less in Oracle VirtualBox than the VMware Workstation and the WSL (Fig. 12.3).

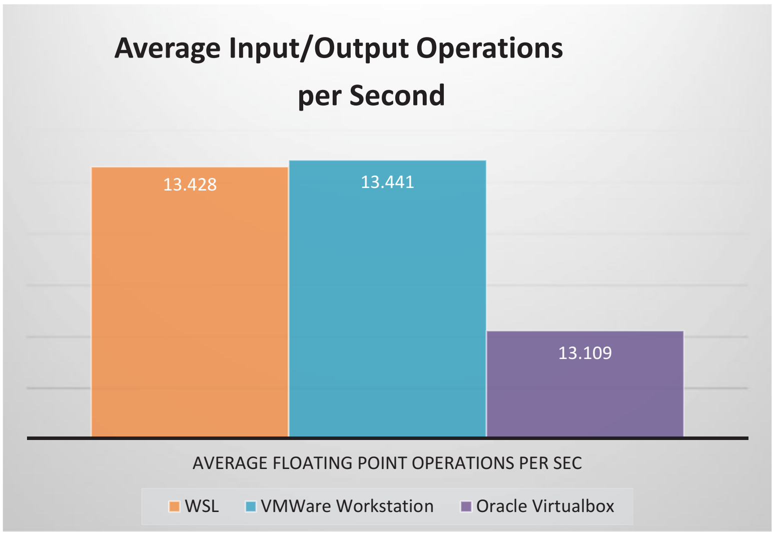

Fig. 12.3 Average I/O operations per second – CPU performance

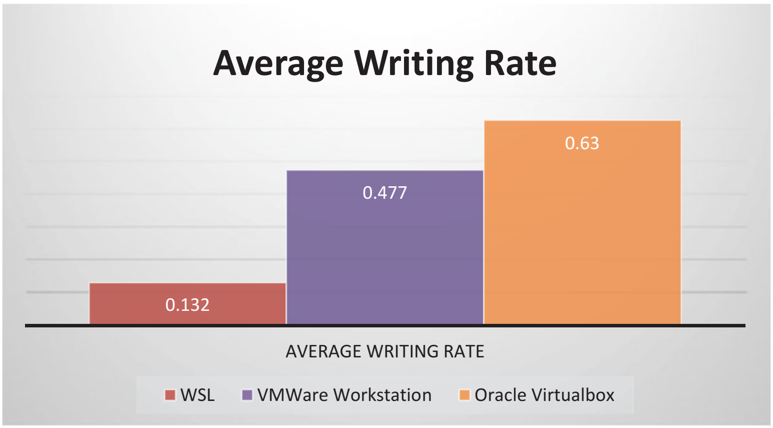

Fig. 12.4 Average writing rate – disk performance

The disk performance is measured by computing the average of the reading and writing rate. The WSL has claimed to have the lowest writing time when compared with the other two virtual boxes (Fig. 12.4).

The WSL reading time creditably has less reading time even though same operations were performed in the other two virtual boxes (Fig. 12.5).

The three virtual environments' overall memory performance and the throughput are calculated for sequential read/write operations and random read/write operations (Fig. 12.6).

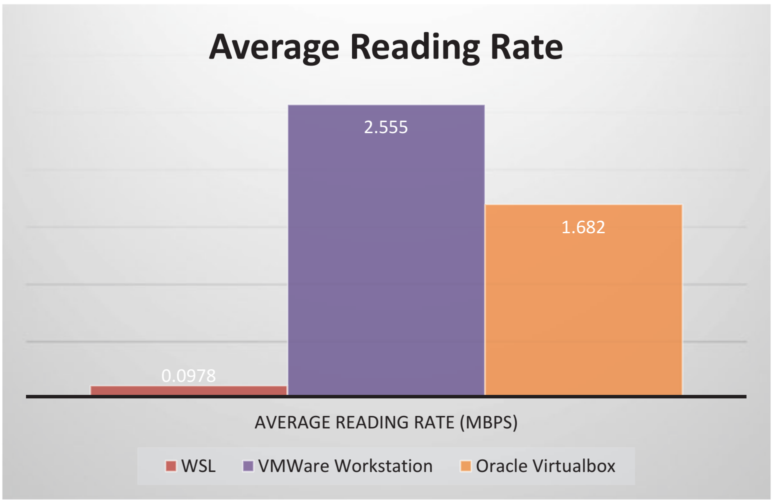

Fig. 12.5 Average reading rate – disk performance

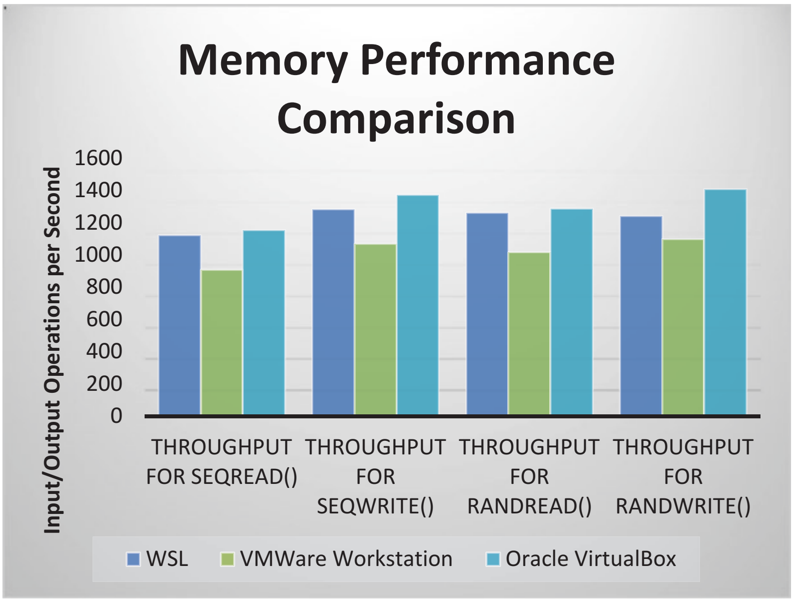

Fig. 12.6 Memory performance

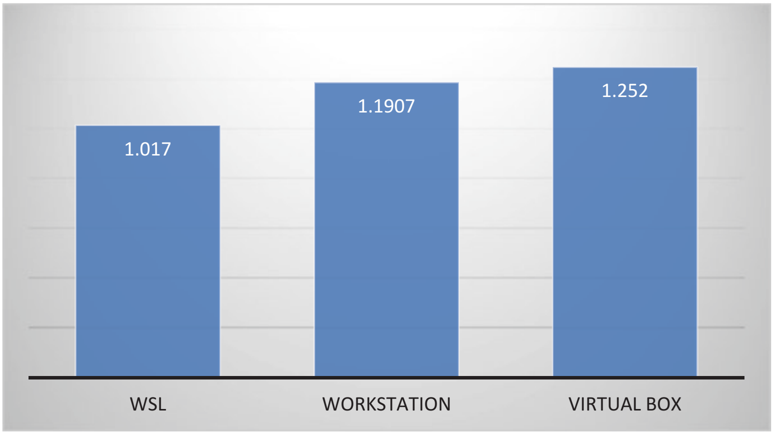

Fig. 12.7 Latency

The latency in the memory performance is measured for all the three virtual boxes, and WSL claims the minimum latency when compared to the other VM boxes (Fig. 12.7).

The average memory performance looks better in Oracle VirtualBox than in the other virtual environments, and the latency is also significantly less. Contrasting over three virtual machines, Oracle VirtualBox has the best memory performance compared to VMware Workstation and WSL. However, WSL tops floating-point operations but is placed second to VMware Workstation in input/output operations in CPU performance. In disk performance, the average writing rate is better in VMware Workstation, but the average reading rate is better in Oracle VirtualBox.

12.6 Conclusion

The performance of the CPU, memory, and disk utilization is considered as the main components in this study. The exhibition of frameworks utilizing Linux as Guest OS is impressively better contrasted with the Windows Host-Guest framework. The experimental results show that each hypervisor has got its own identity. Each one is better than the other in different aspects and can be used according to the enterprise demand and the need for the applications in the virtual environment.

References

1. Alaluna, M., Vial, E., Neves, N., & Ramos, F. M. V. (2019). Secure multi-cloud network virtualization. *Computer Networks, 161*, 45–60. https://doi.org/10.1016/j.comnet.2019.06.004
2. Fischer, A., Botero, J. F., Beck, M. T., et al. (2013). Virtual network embedding: A survey. *IEEE Communication Surveys and Tutorials, 15*, 1888–1906. https://doi.org/10.1109/SURV.2013.013013.00155

3. Nadesh, R. K., & Aramudhan, M. (2018). TRAM-based VM handover with dynamic scheduling for improved QoS of cloud environment. *International Journal of Internet Technology and Secured Transactions, 8*. https://doi.org/10.1504/IJITST.2018.093340
4. Armstrong, D., & Djemame, K. (2011). Performance issues in clouds: An evaluation of virtual image propagation and I/O paravirtualization. *The Computer Journal, 54*, 836–849. https://doi.org/10.1093/comjnl/bxr011
5. Caglar, F., Shekhar, S., & Gokhale, A. S. (2018). ITune: Engineering the performance of xen hypervisor via autonomous and dynamic scheduler reconfiguration. *IEEE Transactions on Services Computing, 11*, 103–116. https://doi.org/10.1109/TSC.2016.2538234
6. Nadesh, R. K., Jagadeesh, M., & Aramudhan, M. (2015). A quantitative study on the performance of cloud data Centre: Dynamic maintenance schedules with effective resource provisioning schema. *Indian Journal of Science and Technology, 8*. https://doi.org/10.17485/ijst/2015/v8i23/51867
7. Bôaventura, R. S., Yamanaka, K., & Oliveira, G. P. (2014). Performance analysis of algorithms for virtualized environments on cloud computing. *IEEE Latin America Transactions, 12*, 792–797. https://doi.org/10.1109/TLA.2014.6868884
8. Nikounia, S. H., & Mohammadi, S. (2018). Hypervisor and neighbors' noise: Performance degradation in virtualized environments. *IEEE Transactions on Services Computing, 11*, 757–767. https://doi.org/10.1109/TSC.2015.2464811
9. Gaj, P., Skrzewski, M., Stój, J., & Flak, J. (2015). Virtualization as a way to distribute PC-based functionalities. *IEEE Transactions on Industrial Informatics, 11*, 763–770. https://doi.org/10.1109/TII.2014.2360499
10. Shea, R., Fu, D., & Liu, J. (2015). Cloud gaming: Understanding the support from advanced virtualization and hardware. *IEEE Transactions on Circuits and Systems for Video Technology, 25*, 2026–2037. https://doi.org/10.1109/TCSVT.2015.2450172
11. Guan, H., Ma, R., & Li, J. (2014). Workload-aware credit scheduler for improving network I/O performance in virtualization environment. *IEEE Transactions on Cloud Computing, 2*, 130–142. https://doi.org/10.1109/tcc.2014.2314649
12. Patikirikorala, T., Wang, L., Colman, A., & Han, J. (2015). Differentiated performance management in virtualized environments using non-linear control. *IEEE Transactions on Network and Service Management, 12*, 101–113. https://doi.org/10.1109/TNSM.2015.2394472
13. Jayasinghe, D., Malkowski, S., Li, J., et al. (2014). Variations in performance and scalability: An experimental study in IaaS clouds using multi-tier workloads. *IEEE Transactions on Services Computing, 7*, 293–306. https://doi.org/10.1109/TSC.2013.46
14. Lakshmi Narayanan, K., & Nadesh, R. K. (2013). Elastic calculator: A mobile application for windows mobile using mobile cloud services. *International Journal of Engineering and Technology, 5*, 2759–2764.
15. Baucas, M. J., & Spachos, P. (2019). Using cloud and fog computing for large scale IoT-based urban sound classification. *Simulation Modelling Practice and Theory, 101*, 102013. https://doi.org/10.1016/j.simpat.2019.102013
16. Nadesh, R. K., & Aramudhan, M. (2018). An empirical study on peer-to-peer sharing of resources in mobile cloud environment. *International Journal of Electrical and Computer Engineering, 8*, 1933–1938. https://doi.org/10.11591/ijece.v8i3.pp1933-1938
17. Crespo, A., Balbastre, P., Simó, J., et al. (2018). Hypervisor-based multicore feedback control of mixed-criticality systems. *IEEE Access, 6*, 50627–50640. https://doi.org/10.1109/ACCESS.2018.2869094
18. Blenk, A., Basta, A., Reisslein, M., & Kellerer, W. (2016). Survey on network virtualization hypervisors for software defined networking. *IEEE Communication Surveys and Tutorials, 18*, 655–685. https://doi.org/10.1109/COMST.2015.2489183
19. Zhu, J., Jiang, Z., Xiao, Z., & Li, X. (2011). Optimizing the performance of virtual machine synchronization for fault tolerance. *IEEE Transactions on Computers, 60*, 1718–1729. https://doi.org/10.1109/TC.2010.224

20. Shea, R., & Liu, J. (2013). Performance of virtual machines under networked denial of service attacks: Experiments and analysis. *IEEE Systems Journal, 7*, 335–345. https://doi.org/10.1109/JSYST.2012.2221998
21. Rajasingh, J. S. J., & Reeves Wesley, J. (2020). Step into the cloud or stop with virtualization - the project manager's dialectic dilemma. *Procedia Computer Science, 172*, 1077–1083. https://doi.org/10.1016/j.procs.2020.05.157
22. Almutairy, N. M., & Al-Shqeerat, K. H. A. (2019). A survey on security challenges of virtualization technology in cloud computing. *International Journal of Computer Science and Information Technologies, 11*, 95–105. https://doi.org/10.5121/ijcsit.2019.11308
23. Jagadeesh, M., & Nadesh, R. K. (2013). Minimization of system resources for cloud based application using web browser add on. *International Journal of Engineering and Technology, 5*, 2777–2784.