İSTANBUL UNIVERSITY PRESS

Acta Infologica
acin 2024, 8 (2), 176–187
DOI: 10.26650/acin.1462076

Research Article

Comparative Analysis of Transaction Performance in Different Virtualization Environments

Ali İhsan Candemir[1], Nilgün İncereis[2]

[1]İstanbul Okan University, Computer Engineering, İstanbul, Türkiye
[2]Bartın University, Distance Education Application and Research Center, Bartın, Türkiye

Corresponding author : Ali İhsan Candemir
E-mail : alcandemir@stu.okan.edu.tr

ABSTRACT

Virtualization technologies are increasing in importance day by day. The selection of virtualization software is an important factor to realize efficient use of physical hardware. In this study, the Ubuntu 22.04 LTS operating system was run on VirtualBox, VMware, and Docker, which are widely used virtualization software. Then, the Tiobench, Compress-7zip, C-ray, Smallpt, Tachyon, and OSBench tests were performed on the Ubuntu operating system with the Phoronix Test Suite software, and the test results were analyzed and compared. The results demonstrate that Docker outperformed the other virtualization technologies, although not in every test. Owing to its rapid deployment and efficient use of resources, Docker is suitable for applications that require agility and scalability. However, traditional virtualization technologies, such as VMware, may be more suitable for applications that require high security and extensive resource management. In addition, this study provides information that can guide users in selecting virtualization software.

Keywords: Virtualization, VirtualBox, VMware, Docker, Virtualization Technologies.

Submitted : 01.04.2024
Revision Requested : 31.05.2024
Last Revision Received : 25.09.2024
Accepted : 04.11.2024
Published Online : 26.11.2024

1. INTRODUCTION

Virtualization technologies include a set of technologies and methods that offer the ability to run multiple operating systems simultaneously by creating virtual environments on physical hardware. These technologies offer advantages such as using resources more efficiently, providing flexibility, and isolating workloads (Schlosser et al., 2011). Virtualization is widely used in many fields, such as cloud computing, server consolidation, application testing, and development (Wei et al., 2019). Virtualization technologies enable more efficient use of physical hardware resources by dynamically sharing them between virtual machines. These technologies are hardware independent and therefore easily portable between different physical environments. They offer security advantages by isolating different virtual machines from each other. Due to these advantages, virtualization technologies continue to be widely used today (Robbi et al., 2019). Processing performance is a critical measure of the speed and efficiency of a system or application's operations. Accurately measuring process performance is important for system optimization, resource allocation, and user experience. Fast processing improves user experience and increases user satisfaction. A good process performance leads to more efficient use of resources. Efficient processes can reduce costs by reducing energy and resource consumption. These items emphasize the importance of processing performance (Mongia et al., 2020).

This study describes virtualization environments at hardware, OS, and application level in section 2, and describes the performance metrics and evaluation criteria in section 3. Then, section 4 describes the materials and methods of this study, and section 5 describes the findings. Finally, section 6 presents the discussion and conclusions of this study.

2. VIRTUALIZATION ENVIRONMENTS

Virtualization environments use various technologies and configurations to support diverse workloads and distribute computing resources efficiently. The three levels of virtualization environments are described below.

2.1. Hardware Level Virtualization

Hardware level virtualization is a type of virtualization that enables the creation of virtual machines on physical hardware. This is implemented using a hypervisor or virtualization manager. Hardware-level virtualization allows each virtual machine to behave as if it had an independent operating system. Each virtual machine can run its own operating system and applications; thus, multiple virtual environments can be achieved on a single physical server. This method offers the advantages of effectively optimizing resource usage and providing isolation (Natawiguna et al., 2016; Perez et al., 2008).

2.2. Operating System Level Virtualization

An operating system-level virtualization scheme allows an operating system instance to host multiple isolated environments. In this type of virtualization, virtual environments (e.g., containers) running on a host operating system are run in isolation. Each container can host its own applications and dependencies. OS-level virtualization is known for its lightweight and fast startup/shutdown processes. Containers share the same core operating system but offer an isolated environment (Perez et al., 2008) (Natawiguna et al., 2016; Patil, 2021).

Containers encompass only the instructions and code required for the application to run; thus, they are lightweight compared to alternative approaches, such as virtual machines (VMs) (Brady et al., 2020).

You can find detailed information about why Docker, which makes container technology useful, is preferred in the study of Ayaz et al., 2016. In addition, Docker is widely used in web applications. (Yılmaz et al., 2016)

2.3. Application-Level Virtualization

Application-level virtualization is a form of virtualization that enables an application to run in an isolated environment within itself. This is typically implemented by running an application in a standalone container or running it isolated in a virtual environment. Application-level virtualization allows applications to run in their own independent operating environments, which offers the advantages of effectively using system resources and ensuring application independence (Perez et al., 2008; Natawiguna et al., 2016).

3. PERFORMANCE METRICS AND EVALUATION CRITERIA

To effectively evaluate virtualization environments, it is important to measure and analyze performance metrics. These metrics are used to evaluate different aspects of virtual infrastructure and the impact of virtual machines on physical resources. The most important performance metrics are CPU utilization, memory management, input/output operations, and network performance.

3.1. CPU use

CPU use refers to the system or application load placed on the processor. Performance metrics were used to determine the length of intensive CPU running. These metrics are used to evaluate the overall effectiveness of CPU utilization, identify performance bottlenecks, and determine whether resources are being used efficiently (Ferrari et al., 1972; Syauqi et al., 2023).

CPU-intensive tests such as c-ray and small, performed using the Phoronix test suite are important for evaluating processor use in a system. The overall efficiency of the CPU was evaluated by measuring the effectiveness of the processor performance in terms of computational tasks.

3.2. Memory Management

Memory management is a critical performance metric that evaluates how an application or system uses and manages memory resources. In this section, we focus on memory utilization, memory leaks, cache memory activity, and memory cleaning strategies. Good memory management can improve system stability and performance (Ferrari, 1972; Syauqi et al., 2023; Sudha, 2013).

Memory intensive operations such as Tiobench and compress7-zip, provide important data for memory management. These tests evaluate the efficiency of memory usage and are particularly useful for studying memory management during on-disk compression and decompression operations.

3.3. Input/Output Operations

Input/output (I/O) operations refer to the ability of an application to interact with a disk, network, or other external resources. Factors such as database access, file read/write speed, and asynchrony of I/O operations were evaluated.

bench testing is an effective way to measure I/O performance. The proposed method analyzes the efficiency of I/O operations by evaluating the performance of file system operations. This is especially important for understanding and optimizing disk performance (Ferrari, 1972; Sudha, 2013).

3.4. Network Performance

Network performance is an important metric that measures the effectiveness of an application or system in transmitting data over a network. In this section, the network bandwidth, latency, packet loss rate, reliability, and network traffic management factors are evaluated. Good network performance positively impacts user experience and ensures that applications run quickly and reliably (Sudha, 2013).

The tachyon test provided by the Phoronix test suite evaluates the network communication performance. This test is useful for measuring the system's network performance in terms of network traffic management, bandwidth use, and reliability.

4. MATERIAL AND METHOD

When analyzing performance, it is important to set up a test environment that can simulate real-life conditions. This increases the validity and reliability of the obtained results and provides decision makers with tangible data.

4.1. Virtualization Platforms for Testing

The virtualization platforms we chose to perform the tests were VirtualBox, VMware and Docker. The virtualization platforms were installed on the same machine, and the Ubuntu 22.04 LTS operating system was run on each machine.

The hardware specifications of the computer used in the tests affect the figures in the test results. However, the performance ratios of virtual machines to each other did not change even when they were tested on different hardware.

The hardware information of the computer on which the virtual machines were installed to perform the tests is described as follows.

Hardware Information
Processor: Intel Core i7-6500U @ 2.50 GHz
Motherboard: Lenovo 20EVS01G00
Memory: 2x8GB SODIMM DDR3 Synchronous 1600 MHz (0,6 ns)
Disk: 500 GB SanDisk SDSSDH3

4.2. Workloads and scenarios

The tests performed using the Phoronix Test Suite on the Ubuntu 22.04 LTS operating system were as follows:

Tiobench: The Tiobench test is used to measure file input/output performance. This test is intended to determine the speed of input/output operations in a file system. In other words, it can be used to measure and compare disk performance. Tiobench was used to test factors such as disk read/write speeds, file processing speeds, and delays in data access (Gurjar et al., 2019).

Compress-7zip: The 7-Zip is an archiving program that provides high compression rates and supports various file formats. The Compress-7zip test in the Phoronix Test Suite measures how fast 7-Zip compresses and archives different file types and sizes (Gupta et al., 2017).

C-ray: This test measures processor performance by simulating 3D modeling operations. It is used specifically to evaluate processor computational capabilities. c-ray shows how effectively processor power is used to render visual effects and 3D modeling using ray tracing. This test was used to measure and compare the performance of the processor cores. It is preferred to see the performance differences between different processor models, core counts, and clock speeds (Benchmark, 2024).

Smallpt: This benchmark test is a 3D ray tracing dataset. This test was used to measure the performance of the ray tracing techniques used to render 3D graphics. Ray tracing is used to create realistic visuals. It is used to simulate how light is refracted and reflected. Smallpt creates 3D scenes using these techniques in a simple manner and measures the rendering time of these scenes (Benchmark, 2024). CPU intensive tests such as c-ray and small, which you run with Phoronix-test-suite, are an important way to evaluate processor utilization in the system. The overall efficiency of the CPU was evaluated by measuring the effectiveness of the processor performance in terms of computational tasks.

Tachyon: It is another ray tracing benchmark test is designed to measure the performance of ray tracing algorithms used to render 3D graphics (Benchmark, 2024).

OSBench: This test aims to measure how fast an operating system can perform basic operating system calls and operations, such as file system access. It is used to evaluate operating system functions, such as file processing, memory management, threading, and other basic system calls (Benchmark, 2024).

For example, the implementation stages of the 7zip compression test on Docker:

First, we pull the Ubuntu 22.04 LTS image from the Docker Hub and run a container. Figure 1 shows the installation and operation of the Docker container.

```
→  ~ docker pull ubuntu:22.04
docker run -it --name phoronix-ubuntu ubuntu:22.04
```

Figure 1. Docker container installation and running

Then, we install the required packages. Figure 2 shows the installation of the required packages.

```
root@5b78ee19f44c:/# apt update
apt install -y software-properties-common wget
add-apt-repository ppa:phoronix-test-suite/public
apt update
apt install -y phoronix-test-suite
```

Figure 2. Installation of required packages

Then, we install the 7zip compression test in the PRONONix test suite Figure 3 shows the installation of 7zip.

```
root@5b78ee19f44c:/# phoronix-test-suite install pts/compress-7zip
```

Figure 3. Install 7zip

As illustrated in Figure 4, the 7zip compression test is conducted.

```
root@5b78ee19f44c:/# phoronix-test-suite run pts/compress-7zip
```

Figure 4. Run 7zip compression test

The results are presented in Figure 5.

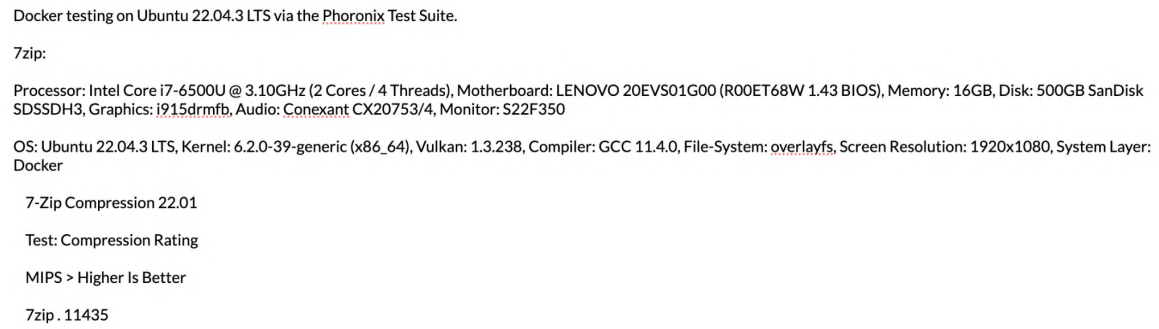

Figure 5. Results

4.3. Data collection and analysis

The test results are presented in tables for clarity and are not presented in the Phoronix Test Suite. In case of inconsistencies in the results, the tests were repeated. Certain variation selections were made in order not to extend the test times. The total test time was measured as 6 h.

Column charts were used to compare and analyze the data made into tables.
Then, the data were analyzed, and comments were made about the usage scenarios.

5. RESULTS

This section presents the results of experimental tests performed in different virtualization environments and their comparative analysis.

5.1. Performance Results of Each Virtualization Environment

The test results are given in the tables below.

5.1.1. Tiobench Test

The tests we apply with the Tiobench test are random write and random-read test.
Test:Size per thread: 64 MB, thread count: 8
The Tiobench test results are presented in Table 1.

Table 1. Tiobench test results

	VirtualBox	VMware	Docker
Random_read (MB/ms)	34,705	37,574	40,638
Random_write (MB/ms)	0032	1159	0255

5.1.2. Compress-7zip Test

7-zip compression and decompression tests were performed. Table 2 lists the results of the Compress-7zip test.

Table 2. Compress-7zip test results

	VirtualBox	VMware	Docker
Compression (MIPS) (million instructions per second)	5557	6495	11435
Decompression (MIPS) (million instructions per second)	5903	5187	8721

5.1.3. C-ray Test

The C-ray and standard ray tracing tests were performed. The results are given in the table below. Test 3 shows the results of the C-ray test.

Table 3. C-ray test results

	VirtualBox	VMware	Docker
Test (ms)	0482	0481	0455

5.1.4. Smallpt Test

The smallpt and standard ray tracing tests were performed. The results are given in the table below. The Smallpt test results are presented in Table 4.

Table 4. Smallpt test results

	VirtualBox	VMware	Docker
Test (ms)	0113	0115	0066

5.1.5. Tachyon Test

The tachyon test and standard ray tracing test were performed. The results are given in the table below. Table 5 presents the Tachyon test results.

Table 5. Tachyon test results

	VirtualBox	VMware	Docker
Test (ms)	2234	1493	1017

5.1.6. OSBench Test

The OSBench test was applied to the thread creation, process creation, and memory allocation tests. Table 6 presents the OSBench test results.

Table 6. OSBench test results

	VirtualBox	VMware	Docker
Create Threads (microsecond/event)	36	57,1	34,7
Create Process (microsecond/event)	85,9	126,2	70,7
Memory allocation (nanosecond/event)	161,9	159,3	136,4

5.2. Comparative Analysis and Findings

The results of each test performed using the Phoronix Test Suite are presented separately using bar graphs for clarity. There were two types of tests we applied in Tiobench tests; random_write and random_read. Because of the random_read test, we found that Docker had the highest read data per second speed among virtual software with a value of 40638 MB/s. In the random_write test, it is noteworthy that the values were very different from each other. Re-tests were performed, and similar results were obtained. This demonstrates that VMware's data-writing speed is superior to other virtualization environments.

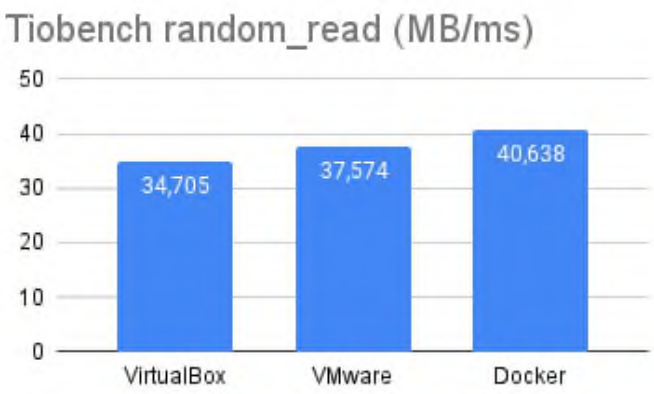

Figure 6. Tiobench random-read test result bar graph.

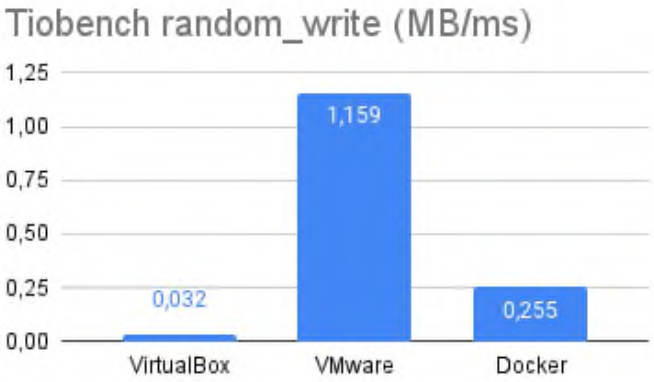

Figure 7. Tiobench random write test result bar graph.

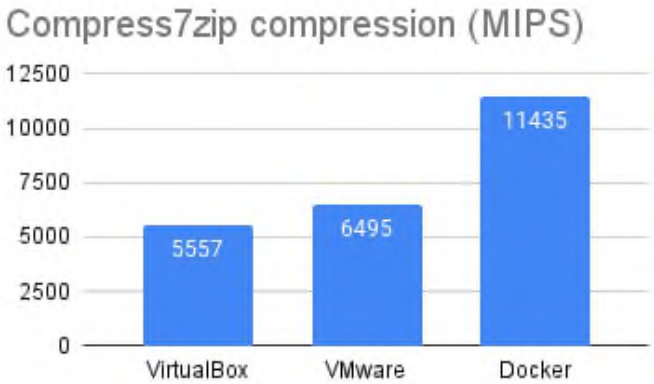

Figure 8. Compress-7zip compression test results are shown as a bar graph.

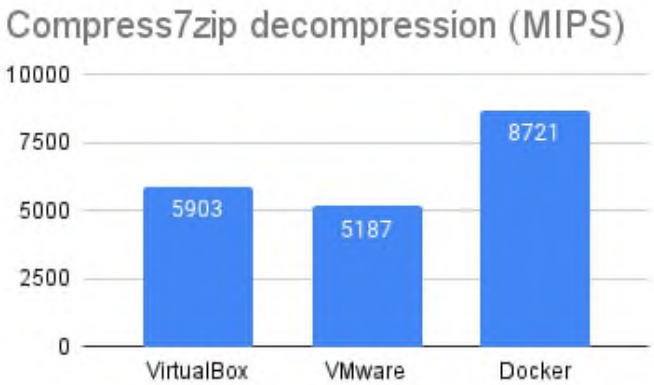

Figure 9. Compress-7zip decompression test results are shown as a bar graph.

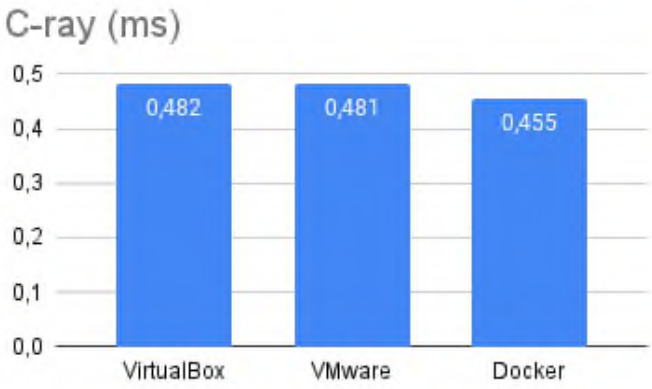

Figure 10. C-ray test results are shown as a bar graph.

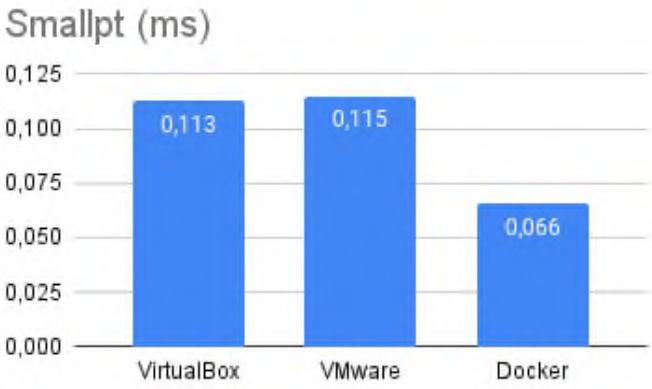

Figure 11. Bar graph of smallpt test results.

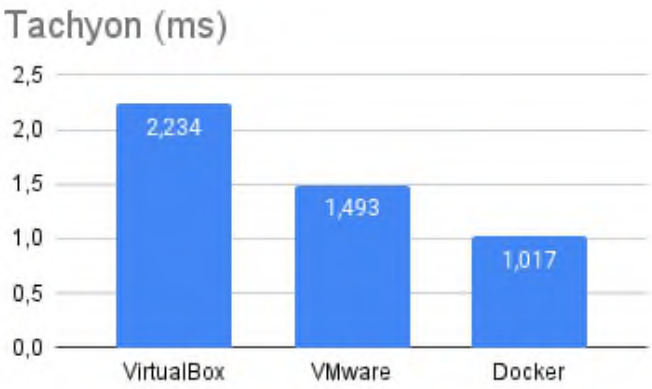

Figure 12. Tachyon test result bar graph.

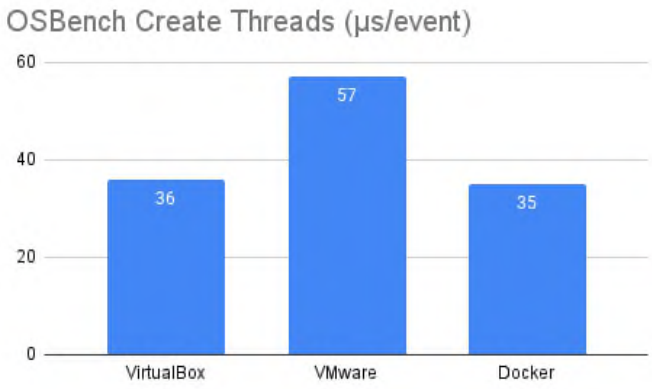

Figure 13. OSBench creates a thread test result bar graph.

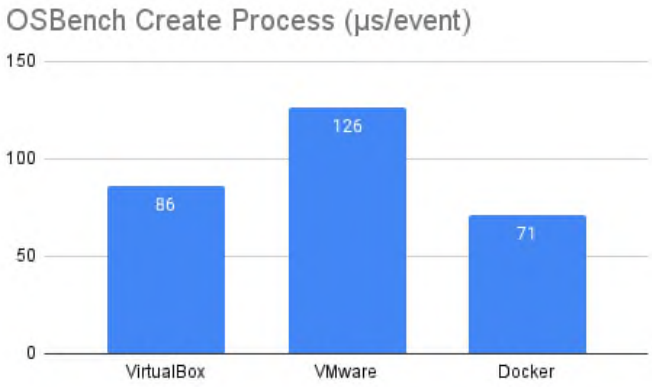

Figure 14. OSBench creates a process test result bar graph.

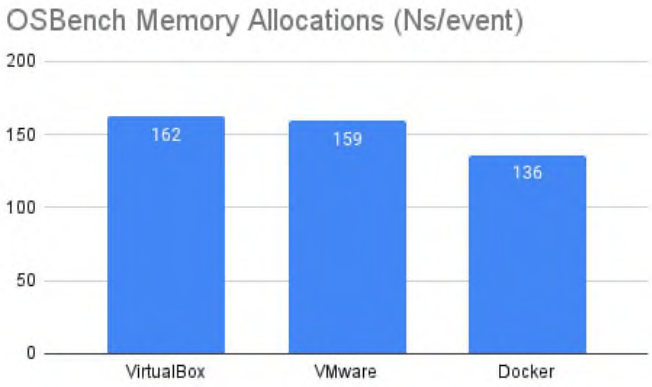

Figure 15. OSBench memory allocation test result bar graph.

In the Tiobench test, the results of which are shown in Figures 6 and 7, it can be seen that Docker demonstrated high performance in the reading test and VMWare in the writing test. This test provides us with information about the disk performance of the virtual machines. In the Compress7zip tests, the results of which are shown in Figures 9 and 10, it can be seen that Docker obtained the highest MIPS (Million instructions per second) values. This demonstrates that Docker outperformed other virtualization environments in tests measuring disk performance. When we look at the C-ray results (Figure 10), which mainly test the processor performance, we see that Docker is superior; however, the results are very close to each other. In the Smallpt (Figure 11) and Tachyon (Figure 12) test results, which focus on processor performance, it is seen that Docker's performance is quite high. However, VMware and VirtualBox demonstrated inconsistent results. This is because these tests measure not only the processor performance. In OSBench's Create Threads (Figure 13) and Create Process (Figure 14) tests; parameters such as processor performance, OS efficiency, memory management, input/output performance, and multi-core usage, were measured. The performance rankings are Docker, VirtualBox, and VMware. Another test from OSBench, Memory Allocations, measures memory-intensive performance such as memory allocation speed, memory-release speed, and memory fragmentation. Although the results (Figure 15) are similar to those of other OSBench tests, VMware appears to perform better than VirtualBox in this test. Docker also demonstrated the highest performance in this test.

Docker takes a different approach from other virtualization technologies. Instead of using hypervisors, as in other virtualization environments, it provides containerization at the operating system level (Pahl et al., 2017; Wang, 2022). This allows Docker containers to be lighter and consume fewer resources (Kaiser et al., 2022; Reis et al., 2022). The fact that MIPS values are superior to those of other virtualization environments suggests that Docker may incur lower overhead (added processing power). This structure of Docker indicates that it will outperform other virtualization technologies in all tests. The results show that this is not the case. We found that Docker outperformed other virtualization environments in certain tests and workloads. In particular, in the Compress-7zip test, we observed that Docker's compression and decompression performance was quite high compared to others. In addition, in the Tiobench test, notably Docker performs quite well in the random read and write operations. However, Docker was not superior in every test. In the Tiobench random write test, Docker outperformed the other virtualization environments.

These results demonstrate that for certain workloads and tests, Docker may have performance advantages; however, these advantages are not always and not in every test scenario. Thus, the selection of a virtualization environment should

depend on the workload, performance requirements, and case. Each can offer different advantages and disadvantages. It is important to choose a virtualization solution that best suits specific requirements (Potdar et al., 2020). VirtualBox and VMware are the most widely used virtualization technologies. Considering the test results, the performance of both virtualization technologies appeared to be close to each other. On average, VMware provides slightly higher performance. When selecting the virtualization technology to use, the use scenario should be considered. Because the usage scenario involves different workloads, performance varies for different workloads.

6. DISCUSSION AND CONCLUSION

Docker's overall success rate in tests was approximately 92%.

Based on the Tiobench test results, Docker achieved the highest read speed of 40,638 MB per second in the random_read test, while it exhibited an average performance of 255 MB in the random_write test. VMwareachievedd the highest write speed (1159 MB. These results demonstrate that Docker exhibits better disk performance, especially in read operations, whereas VMware exhibits better performance in write operations.

In the Compress-7zip test results, Docker achieved the highest values with 11.435 million instructions per second (MIPS) in the compression test and 8.721 MIPS in the decompression test. This demonstrates that Docker is more efficient than other virtualization technologies in terms of disk performance.

Docker obtained the fastest results in the processor-intensive ray tracing test with 455 µs. VMware and VirtualBox gave results of 481 µs and 482 µs respectively. It can be observed that Docker is faster than other processor-based operations, albeit by a small margin.

Dockeroutperformed then other virtualization environments by completing the test in less time than the Smallpt test with 66 µs and Tachyon test with 1017 µs. This demonstrates that Docker is more efficient, especially in CPU-intensive tasks, and provides faster results, especially in 3D graphic operations.

Considering the OSBench test results, Docker obtained the fastest results with 71 µs in the process creation test. VMware and VirtualBox delivered results of 126 and 86 µs, respectively. In addition, Docker demonstrated the fastest performance at 136 ns in the memory allocation test. These findings demonstrate that Docker is more efficient in operational processes such as memory management and process creation.

Docker outperformed other virtualization environments in various performance metrics. In particular, Docker has been observed to be faster and more efficient in tests such as disk reading, compression, ray tracing, and memory management. However, it may have disadvantages, such as security risks and resource isolation. VMware is more suitable for applications that require security and comprehensive resource management and stands out particularly for disk write performance and security requirements. VirtualBox, on the other hand, offered a more balanced performance but fell behind other technologies, especially in terms of disk performance and processor-intensive operations. Users should choose between these virtualization technologies based on their particular needs and application scenarios.

With the increasing use of Docker, security threats targeting Docker containers from malicious hacker teams such as TeamTNT, WatchDog, Kinsing, and Rocke, have increased (Unit42, 2022).

Our findings align with those of Reis and Oliveira (2022), who demonstrated that Docker outperforms traditional virtual machines in terms of start-up time and resource usage. Pahl and Lee (2017) also highlighted that while Docker containers are advantageous for lightweight and scalable deployments, they may not always offer the same level of isolation and security as hypervisor-based solutions. These comparisons underscore the necessity of selecting appropriate virtualization technologies that meet specific workload requirements and performance expectations.

For researchers focusing on virtualization technologies, it is recommended to consider both containerization and traditional virtualization methods depending on the specific case. Docker, with its rapid deployment and efficient resource usage, is ideal for applications requiring agility and scalability. However, for applications needing high security and extensive resource management, traditional virtualization technologies like VMware may be more suitable. Future research should also explore the integration of these technologies to exploit the strengths of both approaches.

This study has certain limitations. First, the tests were conducted using a single type of hardware configuration, which may not represent all possible scenarios. In addition, the focus was on a limited set of performance metrics, thereby eliminating other potentially significant factors, such as long-term stability and maintenance costs. Future studies should consider a broader range of hardware environments and performance indicators.

In future, we plan to consider other factors in the virtualization technology selection process, including security tests. Afterwards, we plan to extend this test, which we conducted using the Ubuntu 22.04 operating system, with different operating systems and other virtualization technologies (Hyper-V, KVM, Parallels).

Peer Review: Externally peer-reviewed.

Author Contributions: Conception/Design of Study- A.İ.C., N.İ.; Data Acquisition- A.İ.C., N.İ.; Data Analysis/Interpretation- A.İ.C., N.İ.; Drafting Manuscript- A.İ.C., N.İ.; Critical Revision of Manuscript- A.İ.C., N.İ.; Final Approval and Accountability- A.İ.C., N.İ.; Technical or Material Support- A.İ.C., N.İ.; Supervision- A.İ.C., N.İ.

Conflict of Interest: The authors have no conflict of interest to declare.

Grant Support: The authors declared that this study has received no financial support.

ORCID IDs of the authors

Ali İhsan Candemir 0009-0001-0167-7879
Nilgün İncereis 0000-0001-5508-8159

REFERENCES

Ayaz, Ö., Aydın, G. (2016), Uygulama Sanallaştırmada Yeni Bir Yaklaşım: Docker. Retrieved from https://ab.org.tr/ab15/bildiri/312.

Brady, K., Moon, S., Nguyen, T., Coffman, J. (2020), *Docker Container Security in Cloud Computing*, 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), doi: 10.1109/CCWC47524.2020.9031195

Benchmark Tests: https://openbenchmarking.org/tests, 01.01. 2024.

Ferrari, D. (1972). *Workload characterization and selection in computer performance measurement*. Computer 5: 18-24.

Gupta, A., Bansal, A. and Khanduja, V. (2017), *Modern lossless compression techniques: Review, comparison and analysis,* 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, pp. 1-8.

Gurjar, D. and S. S. Kumbhar, (2019) *A Review on Performance Analysis of ZFS & BTRFS*, 2019 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, pp. 0073-0076, doi: 10.1109/ICCSP.2019.8698103.

Kaiser, S., Haq, M.S., Tosun, A.S., & Korkmaz, T. (2022). Container technologies *for ARM Architecture: A Comprehensive Survey of the State-of-the-Art*, IEEE Access, 10, 84853-84881.

Mongia, V., Sharma, A. (2020 July-August), *Performance Comparison of Virtual Machine Selection Policies in Cloud Environment,* International Journal of Advanced Trends in Computer Science and Engineering, Volume 9, No.4.

Natawiguna, A., & InggrianiLiem, M. M. (2016), *Virtualization Methods for Securing Online Exam*, 2016 International Conference on Data and Software Engineering (ICoDSE), Denpasar, Indonesia, pp. 1-7.

Pahl, C., Brogi, A., Soldani, J., & Jamshidi, P. (2017). Cloud container technologies: *A State-of-the-art review. IEEE* Transactions on Cloud Computing, 7, 677-692.

Patil, S. (2021 May), *Study of container technology with docker.* International Journal of Advanced Research in Science Communication and Technology.

Perez, R., vanDoorn, L., & Sailer, R. (2008 November). *Virtualization and hardware-based security*. IEEE Security and Privacy Magazine 6(5):24 – 31

Potdar, A.M., Naraya, D. G., Kengond, S., Mulla, M.M. (2020), *Performance Evaluation of Docker Container and Virtual Machine*, Procedia Computer Science, 171, 1419-1428.

Reis, D., Piedade, B., Correia, F. F., Dias, J. P., & Aguiar, A., (2022). *Developing Docker and Docker-Compose Specifications: A Developers' Survey.* IEEE Access, vol. 10, pp. 2318-2329.

Robbi, F. A. A., A. B, Prasetijo, and E. D., Widianto ED (2019 January), *Perbandingan Kinerja Block Storage Ceph dan ZFS di Lingkungan Virtual*, Jurnal Teknologi dan Sistem Komputer, vol. 7, no. 1, pp. 7-11.

Schlosser, D., Duelli, M., & Goll, S. (2011). Performance comparison of hardware virtualization *Plaforms*, International Conference on Research in Networking, 393-405.

Sudha, M. (2013), *Performance Analysis of Kernel-Based Virtual Machine*, International Journal of Computer Science and Information Technology.

Syauqi, M. R., Sabrina, N. P., Santikarama, I. (2023), *K-Means Clustering with KNN and mean computation on CPU Benchmark Compilation Data*, Journal of Applied Informatics and Computing, 7(2), pp. 231–239. https://doi.org/10.30871/jaic.v7i2.6491

Unit42 (2022, April 12). I Am Your Defense against Cloud Threats: The Latest Unit 42 Cloud Threat Research. Available online: https://unit42.paloaltonetworks.com/iam-cloud-threat-research (accessed on 20 July 2024).

Wang, W., (2022), *Research on Using Docker Container Technology to Realize Rapid Deployment Environment on Virtual Machine*, 2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC), Hangzhou, China, pp. 541-544.

Wei, S., Zhang, K., & Tu, B. (2019 June). *HyperBench: A benchmark suite for virtualization capabilities*. Proc. ACM Meas. Anal. Comput. Syst. 3, 2, Article 24, 22 pages.

Yılmaz, E. C., Oktaş, R., & Karabulut, B. (2016). DOCKER İLE WEB UYGULAMASI GELİŞTİRME. Küresel Mühendislik Çalışmaları Dergisi, 3(2), 116-121.

How cite this article

Candemir, A. İ., & İncereis, N. (2024). Comparative Analysis of Transaction Performance in Different Virtualization Environments. *Acta Infologica, 8*(2), 176-187. https://doi.org/10.26650/acin.1462076