Malaysian Journal of Computing, 9 (2): 1838-1851, 2024

eISSN: 2600-8238

A NEW FRAMEWORK FOR DEPLOYING VIRTUAL DESKTOP INFRASTRUCTURE USING CONTAINERIZATION APPROACH

Nukman Samsuddin [1], Mohamad Yusof Darus[2*] and Muhammad Azizi Mohd Ariffin

[1,2,3]College of Computing, Informatics and Mathematics*
Universiti Teknologi MARA
[1]nukman_18697@yahoo.com, [2*]yusof_darus@uitm.edu.my, [3]mazizi@fskm.uitm.edu.my

ABSTRACT

The COVID-19 pandemic accelerated the shift to virtual learning, pushing educational institutions to adopt innovative solutions like Virtual Desktop Infrastructure (VDI). However, traditional VDI faces performance issues due to resource limitations. This research aims to enhance VDI performance and scalability by proposing a new framework based on containerization to address performance degradation issues such as unutilized CPU and memory usage in VDI. Four key performance metrics—CPU performance, memory performance, boot time, and latency—were evaluated. This research employed a qualitative method involving five phases: problem identification, planning, design and development, testing, and result analysis. The results show that the proposed containerization framework offers superior performance, with the proposed container standing out as the most efficient platform. For instance, the average boot time is merely 10 seconds compared to Windows' 39 seconds. Regarding RAM usage, the proposed container uses 7%, compared to 27% for Windows and 33% for Linux. This research highlights the potential of the proposed framework to revolutionize virtual learning by optimizing resource utilization, enabling educational institutions to support larger student populations with high-performance standards. It contributes to the ongoing efforts to optimize IT infrastructure for virtual learning, addressing key performance challenges.

Keywords: *Cloud Computing, Online Learning and Framework, Virtual Desktop Infrastructure (VDI), Virtual Machines Containerization.*

Received for review: 08-05-2024; Accepted: 26-08-2024; Published: 01-10-2024
DOI: 10.24191/mjoc.v9i2.25672

1. Introduction

COVID-19, the outbreak that began in late 2019 and quickly expanded over the world. Many countries in a worldwide state of emergency responded to the outbreak of COVID-19 by implementing important decisions on the ground, including social distancing, curfews, lockdowns of cities, and the closing of schools and universities (Affouneh *et al.,* 2021). The desire to return to conventional face-to-face learning has received a lot of attention. The unexpected and unplanned shift to virtual classrooms and online learning in early 2020 challenged those working in educational environments. The transition from classroom instruction to completely online learning was unprecedented (Brown *et al.,* 2021). Furthermore, factors such as learning facilities are crucial for the educational process and play a significant role in shaping the learning experience.

The success of online learning relies heavily on the quality of students' learning experiences. It is essential to create a virtual environment that mirrors the on-campus setting, providing students with access to academic resources, tools, and crucial software (Rodríguez Lera, 2021). One challenge in virtual learning is the limited capability of students' devices to run certain software with higher CPU requirements, such as Oracle Database and MATLAB. A solution to this challenge is the adoption of VDI through cloud-based virtualization services. Transitioning to cloud technology and implementing VDI for virtual learning necessitates a carefully planned strategy that considers the institutional environment and IT infrastructure investments.

Nevertheless, the current challenge with VDI lies in its nature as a form of desktop virtualization on the cloud, where specific desktop images operate within VMs and are transmitted to end clients via a network (Dong, 2021; Nian, 2021). The virtualization software itself demands resources, limiting the VM's available resources. Instead of dedicating an entire machine, VMs can allocate resources more closely aligned with their actual task requirements, thereby reducing the risk of resource underutilization. One effective approach to addressing this issue is through Containerization, in contrast to VMs. Containerization involves constructing software containers as application packages from individual images, providing lightweight virtualization that consumes fewer resources and less time. Containers are more efficient and quicker to create and migrate compared to VMs, occupying fewer system resources such as memory and disk space. This research aims to comprehensively address the existing state-of-the-art research on the aforementioned challenges in VDI. Furthermore, the transition to a VDI leveraging containerization necessitates the development of a comprehensive framework. This framework is essential to provide structured guidelines for effectively managing containerized VDI environments, ensuring optimal resource utilization, performance, and security. By integrating containerization into VDI, educational institutions can augment accessibility and flexibility while overcoming challenges inherent in traditional virtualization methods.

Therefore, this research proposes a new framework for VDI based on the concept called Containerization to improve the degradation of performance in VDI. The new framework will then be implemented, tested, and evaluated in the same environment as VDI. Containerization is a standard method of packaging an application's code, runtime, system tools, system libraries, and configurations into one instance. Cloud computing uses it to construct blocks that contribute to operational efficiency, version control, developer productivity, and environmental consistency. As a result of these considerations, users can expect reliability, consistency, and speed irrespective of the distributed platform. This enhanced infrastructure affords greater control over fine-grained resource management. The use of containers in online services also contributes to improved storage, bolstering cloud computing's information security, availability, and scalability.

2. Literature Review

Cloud computing refers to the practice of storing and accessing data and programs over the Internet rather than on our computer's hard drive. The Internet itself is symbolically depicted as a cloud in the context of computer networks (Rashid & Chaturvedi, 2019). The key driving forces behind cloud computing are the ubiquity of broadband and wireless networking, falling storage costs, and progressive improvements in Internet computing software. Cloud-service clients will be able to add more capacity at peak demand, reduce costs, experiment with new services, and remove unneeded capacity, whereas service providers will increase utilization via multiplexing, and allow for larger investments in software and hardware.

Meanwhile, virtualization has fundamentally reshaped the landscape of cloud computing. It involves creating virtual representations of servers, operating systems, storage, networks, and application resources, leading to reduced machine hardware costs and energy consumption (Kumar *et al.,* 2021). The concept of virtualization dates back to the 1960s, when IBM pioneered the creation of virtual machines for accessing replicated interfaces in mainframe computers. Over time, advancements in virtualization technology have significantly improved virtual desktops. Presently,

virtual desktop solutions are primarily categorized into two types: VDI and Server-Based Computing (SBC) (Wan *et al.*, 2020).

Figure 1 shows the commonly used hypervisor and VDI structure, where each user requests a login connection for a virtual desktop. The user request is proceeded to Connection Management System (CMS). The CMS takes the login request information and passes it to the Authentication Management System (AMS). The CMS and AMS are very important for this whole process because they are involved in validating user authentication and delivering the virtual desktop to the requested user. Once the user gets successfully authenticated from the AMS, the CMS initiates a request to the hypervisor to allocate a virtual desktop for the requested user. When the virtual desktop is assigned to the requested user from the hypervisor, the CMS immediately delivers that virtual desktop to the requested user. Subsequently, the user can use the virtual desktop as it is a personal desktop. The storage devices are used for storing the VMs. The roles of Hypervisor Management System (HMS) are used to manage the hypervisor.

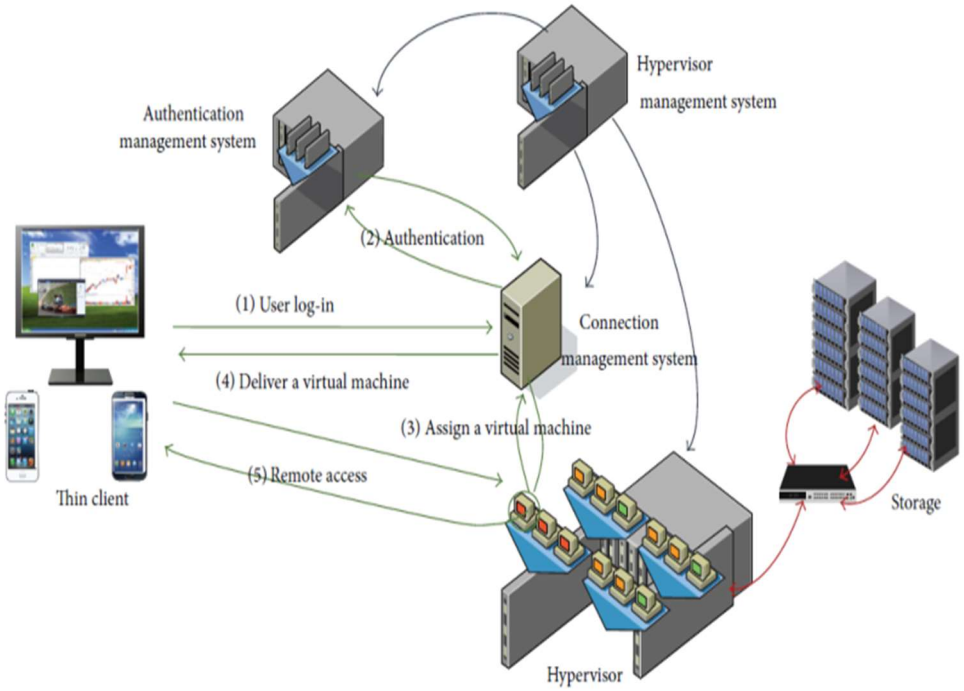

Figure 1. Illustration of the concept of Virtualization (Rahman *et al.*, 2019)

The primary goal of virtualization is to transform traditional computing methods to enhance scalability, efficiency, and cost-effectiveness. Substantial efforts are underway in this field to streamline and manage computing workloads more effectively. Through the utilization of VDI, depicted in the illustration above, authorized users have the flexibility to access their virtual desktop workspace anytime and from anywhere via internet connectivity. The hypervisor plays a crucial role in this process by creating distinct Virtual Machines (VMs), each serving as a virtual desktop. Resources such as memory, operating system, CPU, network, and data are shared among various virtual desktops and are entirely managed by the hypervisor.

Hypervisors are classified into two different types: Type-1: native or bare-metal hypervisors and Type-2: hosted hypervisors. Type-1 hypervisors run directly on the host machine's physical hardware called bare-metal hypervisors. It is installed directly on top of the physical server's hardware, there is no operating system or any other software layer in between. Type-1 hypervisors are actually a very basic OS on top of which we run virtual machines. So, the physical machine on which the hypervisor is running can only be used for virtualization purposes and nothing else.

The advantages of Type-1 hypervisors are they are not constrained by the inherent limitations that come with OSes, and hence can provide great performance. Also, since Type-1 run directly on the physical hardware without any underlying OS, they are secure from the flaws and vulnerabilities that are often endemic to OSes. This ensures that every VM is isolated from any malicious software activity.

Type-2 hypervisors run on the operating system of the physical host machine, hence they are also called hosted hypervisors. These hypervisors are hosted on the OS, and the hypervisor runs on that layer as another software to enable virtualization. These hypervisors are usually used in environments where there are a small number of servers. They do not need a separate management console to set up and manage the virtual machines. These operations can typically be done on the server that has the hypervisor hosted. This hypervisor is basically treated as an application on your host system.

Container or containerization technology is a method of packaging an application so that it can run with isolated dependencies, and it has fundamentally changed software development today due to computer system compartmentalization (Basyildiz, 2019). Container technology was created in 1979 with Unix version 7 and the chroot system. The chroot system isolates a process by restricting an application's access to a specific directory, which comprised of a root and child directories.

Containers are lightweight software components that bundle the application, its dependencies, and its configuration in a single image, running in isolated environments (applications, libraries and binaries) on a traditional operating system on a traditional server or in a virtualized environment as shown in figure 2.7. Isolation in context means quick and responsive; containers are smaller entities than virtual machines, allowing them to be deployed much faster and with shorter startup times. Containers are now widely used in development and web services, and they are gaining traction in the burgeoning field of data science (González & Evans, 2019). In this scientific research environment, the allure of containers is clear: to program all the software libraries necessary in VDI to be placed in a container and distributed. This is to ensure the improvement of VDI performance, and the programs installed will run consistently across all computers and servers in the laboratory, whether it is on a student's Windows laptop or a departmental Linux server. Containers will be the basis of the proposed framework for this project to improve the performance of VDI. Frameworks typically include reusable components, libraries, and APIs that streamline development processes and promote best practices and often embody design patterns, architectural principles, and coding conventions that help developers build scalable, maintainable, and secure applications.

The definitional framework unifies the traditional technology focus of the definitions and integrates additional elements that are likely to increase the adoption of a comprehensive definition to support the development of future business applications. Furthermore, the framework serves as a reference set for scholarly societies and standards organisations in the future and the framework is intended to provide guidance when researchers want to evaluate how existing or proposed legal, economic and/or policy models will work when confronted with the socio-technical change brought about by these technologies (Manwaring & Clarke, 2015).

3. Proposed Framework

This paper suggests a new framework utilizing containerization methods to address performance issues, including underutilized CPU and memory in VDI. Figure 2 depicts an intricately designed infrastructure tailored to facilitate remote access for students or users via the internet. Comprising two fundamental nodes, namely the Virtual Network Computing (VNC) Server node and the Controller Node. The proposed infrastructure is characterized by its potential deployment across multiple servers, offering both redundancy and scalability.

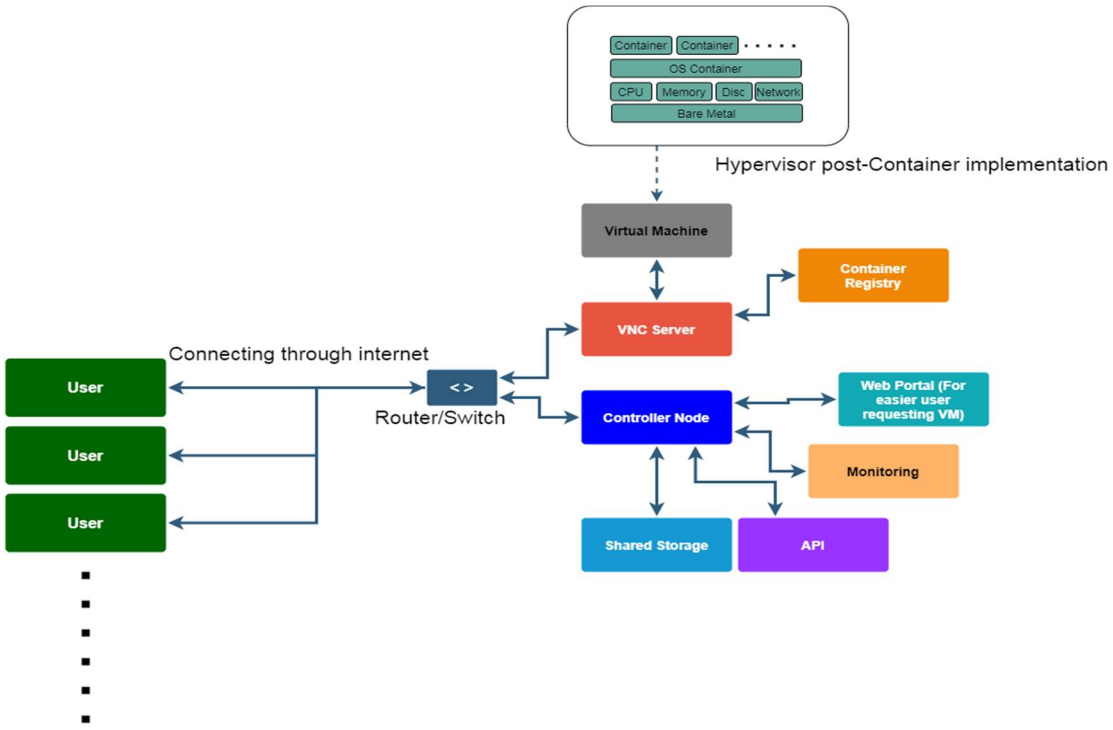

Figure 2. The Proposed Framework

3.1 VNC Server Node

a. **Functionality**: The VNC Server node operates as a sophisticated screen-sharing system, functioning seamlessly across diverse platforms. Its primary objective is to enable users to remotely control a computer from virtually any geographical location, utilizing the computer's screen, keyboard, and mouse independently. This functionality is crucial for remote work, technical support, and collaborative projects, allowing users to access and manage their systems as if they were physically present. By providing a reliable and secure connection, the VNC Server node enhances productivity and flexibility, supporting a wide range of use cases across various industries and applications.

b. **Components**:

- ***Virtual Machines****:* This node hosts virtual machines that users can access remotely using various remote display protocols, such as Remote Desktop Protocol (RDP), VMware Blast, or Citrix HDX. These protocols transmit the graphical user interface and user inputs between the client device and the virtual machine, ensuring a seamless and interactive experience. Each VM represents a fully virtualized desktop environment, complete with its own operating system, applications, and user configurations, allowing users to work in a consistent and personalized workspace. This setup facilitates a high level of flexibility and efficiency, as users can access their virtual desktops from any location, using any compatible device.
- ***Container Registry****:* This includes a Container Registry designed for the systematic management of container images. Users can upload (push) images to the registry and download (pull) them onto other systems, facilitating the

execution of those systems based on the pulled images. This streamlined process ensures that container images are efficiently managed and easily accessible, promoting seamless deployment and consistency across different environments.

c. **Containerization**: Allows the deployment of applications within isolated containers. These containers encapsulate all the necessary components, including libraries and dependencies, required for the application to run. This isolation ensures that each container operates independently without interference from other containers or the underlying host system. Containerization will be applied on the hypervisor; hence the post container implementation will be hypothetically the same as the schematic in Figure above.

3.2 Controller Node

a. **Functionality**: The Controller Node assumes a pivotal role in orchestrating communication and interaction among various services, applications, and platforms inherent to the infrastructure.

b. **Components**:
   - ***Web Portal****:* A dedicated website is provided for users to request virtual machines through an accessible web interface.
   - ***Monitoring****:* This integral component facilitates the monitoring of servers, virtual machines, and network performance through the utilization of the Hypertext Transfer Protocol (HTTP).
   - ***API (Application Programming Interface)****:* The infrastructure integrates interfaces designed to elucidate the operational framework for users, administrators, and potential administrators, thereby facilitating comprehension of the infrastructure's functioning. APIs serve as conduits for diverse software components to communicate and engage in interactions.
   - ***Shared Storage****:* Memory resources that are accessible and shared among multiple servers, specifically configured for storing files, particularly those of significant quality and large size.

The infrastructure is meticulously architected to afford remote access to virtual machines through a VNC Server node with its orchestration and management being vested in the Controller Node. While the VNC Server ensures remote control and screen-sharing functionalities, the Controller Node governs communication, monitoring, and judicious resource sharing within the infrastructure. The amalgamation of virtualization, containerization, and a web-based portal collectively begets a versatile and easily accessible environment for users.

4. Result and Analysis

In the testing phase, **four (4)** performance metrics are assessed to evaluate the effectiveness of the currently used virtual machines. These metrics include boot time, RAM usage, CPU usage, and latency (Ahmadi, 2013; Ahmed *et. al*., 2023; Auliya *et. al*. 2024). During periods of inactivity, examining RAM and CPU usage provides insights into the baseline resource consumption, aiding in the identification of any unnecessary background processes or services. Concurrently, when software is actively running, the testing process gauges its impact on computer resources, such as

RAM and CPU, elucidating the additional load on the system. This analysis facilitates the optimization and efficient utilization of resources.

The evaluation involves two virtual machines and a Linux Container (LXC container), all configured with identical virtual hardware specifications—specifically, a four-core CPU and 4GB of total RAM. It is important to note that the two virtual machines are set up to host different operating systems: Windows 10 and Linux Server, respectively.

4.1 Boot Time

Boot time is a common benchmark that is used to measure the performance of a bootable device. Longer boot times over the course of the system's life can be a sign of issues like malware, device conflicts, and inefficient configuration. This performance benchmark will also help to measure the issue of performance degradation in the system and the applications and services running on it

This benchmark test is performed by booting, shutting down and then rebooting virtual machines for **five (5)** times to determine the average boot time for both used operating systems as shown in Figure 3.

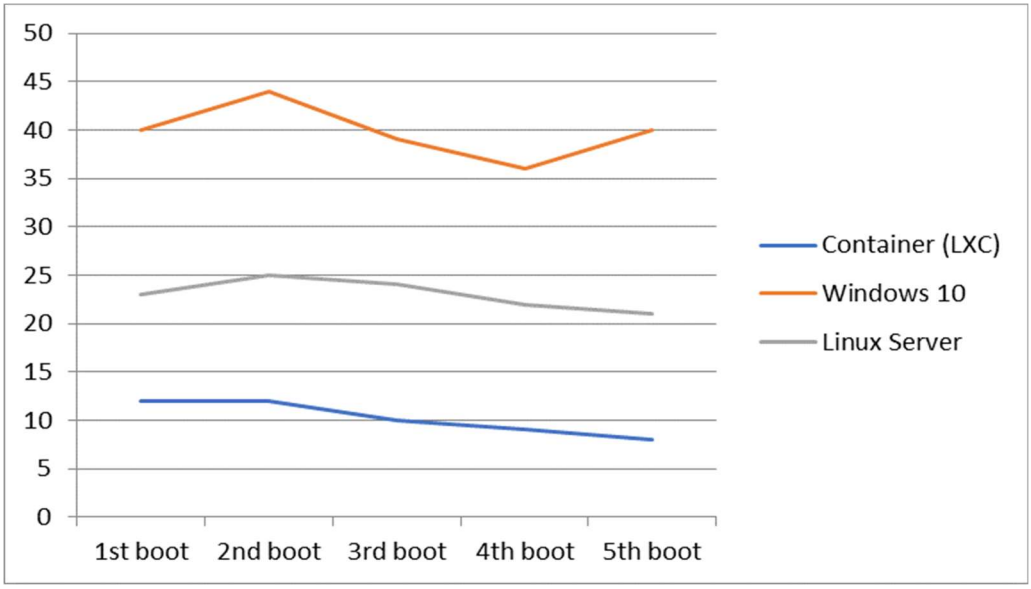

Figure 3. Testing for Boot Time.

Figure 3 and Table 1 indicate that the boot time average for Windows 10 in VDI is 39 seconds, while for the Linux operating system, it is 23 seconds. Using containers, the boot time is only 10 seconds. The test results demonstrate that the utilization of containers accelerates the boot time in VDI.

Table 1. Testing for Boot Time.

Boot Time (in Seconds)						
	1st boot	**2nd boot**	**3rd boot**	**4th boot**	**5th boot**	**Average**
Container (LXC)	12	12	10	9	8	10
Windows 10	40	44	39	36	40	39
Linux Server	23	25	24	22	21	23

4.2 Memory Utilization

Memory utilization is a crucial factor in delivering a responsive and efficient desktop experience in VDI. It is imperative to allocate, monitor, and manage memory properly to ensure sufficient resources for the required number of virtual desktops, applications, and workloads.

In this research, testing is conducted in two ways. Firstly, the VMs are kept idle, with no software running except for default background processes executed during and after boot time. This idle state is maintained for five hours, and memory usage is recorded at one-hour intervals. Secondly, a separate test involves running a 2D platformer game, Mario, on each VM, and the RAM usage of each VM is recorded during this activity.

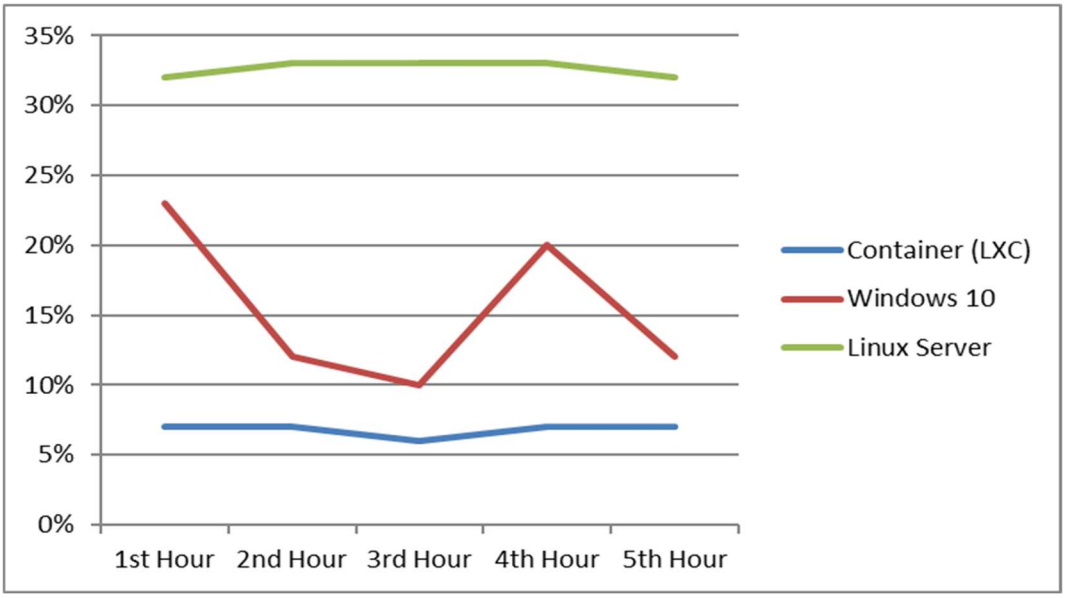

Figure 4. Testing for Memory Utilization

Table 2. Testing for Memory Utilization

RAM Usage (idle)					
	1st Hour	**2nd Hour**	**3rd Hour**	**4th Hour**	**5th Hour**
Container (LXC)	7%	7%	6%	7%	7%
Windows 10	23%	12%	10%	20%	12%
Linux Server	32%	33%	33%	33%	32%

As shown in Figure 4 and Table 2, Linux servers often exhibit higher RAM usage during idle periods compared to Windows 10. This is due to differences in memory management strategies. Linux prioritizes memory caching and buffering to optimize performance, utilizing RAM for caching frequently accessed data and buffering I/O operations. This aggressive caching strategy enhances system responsiveness but results in higher RAM usage during idle. High RAM usage on a Linux server can have a negative impact on VDI performance. Insufficient available memory can lead to performance degradation, including slower response times, increased latency, and potential desktop freezing. Swapping and paging, which occur when RAM is under pressure, introduce additional latency and disk I/O operations, further reducing VDI performance.

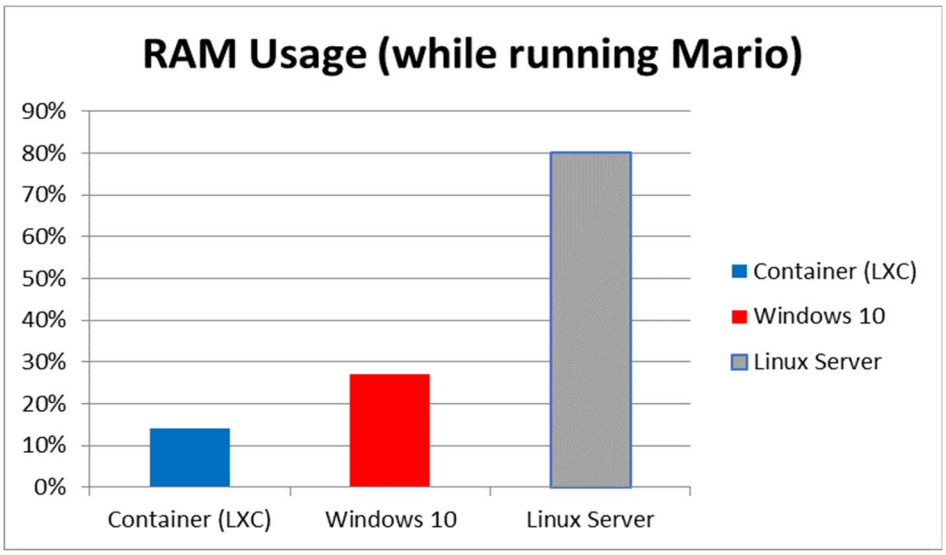

Figure 5. Testing for Memory Utilization (Mario Application)

Table 3. Testing for Memory Utilization (Mario Application)

Platform	RAM Usage (while running Mario)
Container (LXC)	14%
Windows 10	27%
Linux Server	80%

As depicted in Figure 5 and Table 3, executing applications such as "Mario Infinite" on a freshly installed Linux server with a 4-core CPU and 4GB RAM and comparing it to running the same game on Windows 10 with identical hardware specifications can offer valuable insights into the differences in RAM usage between the two systems.

The testing results indicate a notable discrepancy in RAM usage between the Linux server and Windows 10 when running the "Mario Infinite" application on Google Chrome. This suggests that, in this specific scenario, the Linux server may have utilized more RAM compared to Windows 10. Interestingly, the container exhibited efficient resource management by utilizing only 14% of the RAM memory during the execution of the "Mario Infinite" application.

4.3 CPU Utilization

CPU utilization, much like RAM, plays a vital role in VDI. Each virtual desktop requires CPU resources to function effectively. It's crucial to allocate enough CPU power to ensure responsive user experiences. The number of users and workload intensity both have an impact on CPU usage. Higher user density or demanding workloads increase CPU demand, potentially leading to performance issues.

Figure 6 and Table 4 illustrate that the Linux server exhibited higher CPU usage (50%) compared to Windows 10 (14%). This disparity can be attributed to the way operating systems schedule and prioritize processes, which significantly impacts CPU usage. Linux, known for its efficient process scheduler, may distribute CPU resources differently than Windows 10, leading to elevated CPU usage on the Linux server.

This discrepancy in process scheduling could pose challenges for VDI, particularly since Linux servers are commonly utilized for hosting websites. The hosting of multiple websites can consume substantial server resources, including CPU, memory, and disk space. If a considerable portion of the server's resources are dedicated to hosting websites, it might restrict the available resources for other VM deployments. Consequently, this allocation imbalance could result in diminished performance or scalability for virtual desktops.

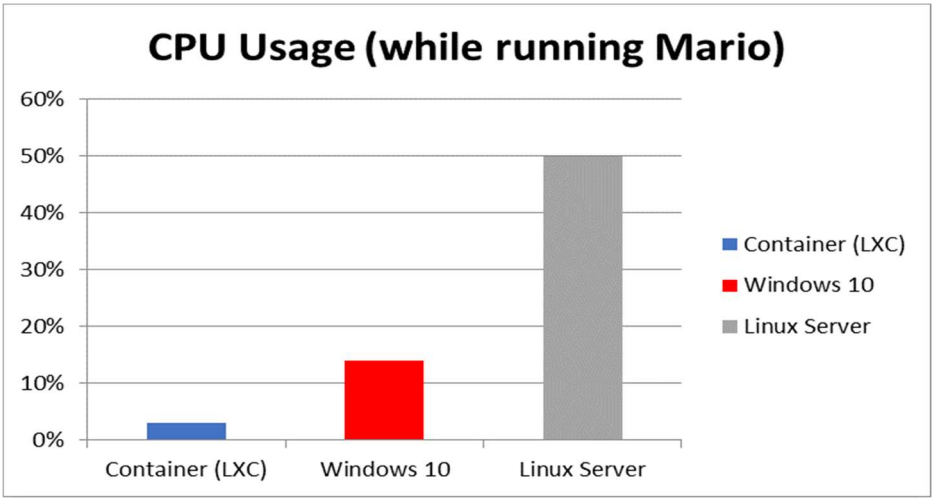

Figure 6. Testing for CPU Utilization

Table 4. Testing for CPU Utilization

Platform	CPU Usage (while running Mario)
Container (LXC)	3%
Windows 10	14%
Linux Server	50%

4. 4 Latency

Latency, in the context of VDI, pertains to the delay or response time between a user's action and the corresponding system response. It plays a pivotal role in shaping the user experience and overall system performance. To assess the impact of different operating systems on network performance, a 'Frame Delay' test will be conducted using the same software, Mario Infinite, with a consistent 'frames per second' (FPS) configuration set at 30 FPS.

The 'Frame Delay' test, also known as input lag or display lag, measures the delay between a user's input, such as pressing a button or moving a joystick, and the subsequent action or change displayed on the screen in a video game. This delay directly influences the responsiveness and real-time interaction between the player and the game. The test involves pressing a button or moving the joystick on the controller, followed by measuring the time it takes for the corresponding action to be displayed on the screen (in frames, F).

The table reveals that both Windows 10 and the Linux server demonstrate a consistent 3-frame (0.0333 seconds) input delay when playing Mario Infinite. This suggests that both operating systems deliver a comparable level of responsiveness in terms of processing user inputs and rendering corresponding actions on the screen. Consequently, latency is not anticipated to be a significant issue in VDI, at least within a short-time span.

Table 5. Testing for Latency

Platform	Latency (frame delay (F)
Container (LXC)	3
Windows 10	3
Linux Server	3

However, it's crucial to note that latency is influenced by the overall performance of the server. Latency and network traffic share an interconnected relationship in the context of network communications. Hosting multiple websites and launching numerous VMs can result in heightened network traffic, increased disk I/O, and elevated CPU usage. If these activities vie for system resources alongside VDI workloads, it may lead to performance degradation and reduced responsiveness for virtual desktop users.

5. Conclusion

This research contributes to ongoing efforts to optimize IT infrastructure for virtual learning by addressing critical performance challenges associated with traditional VDI and proposing an innovative containerization framework, offering valuable insights and practical solutions for educational institutions navigating remote education complexities and enhancing the overall learning experience. The findings not only provide immediate benefits in improved performance and resource efficiency but also lay the groundwork for future advancements in virtual learning technologies, shaping the future of virtual learning environments as educational institutions adapt to the changing landscape of education.

The implementation of this framework in server environments presents several noteworthy advantages. Through encapsulating applications and their dependencies within lightweight, portable containers, resource utilization becomes more efficient, fostering enhanced scalability and flexibility. The isolation provided by containers ensures that applications can operate independently, minimizing potential conflicts and optimizing the overall impact on server performance. This positive impact on resource utilization makes this framework a valuable and increasingly prevalent approach in modern IT infrastructures.

5. 1 Limitations

Containers are typically designed to be ephemeral and stateless, meaning they do not retain any data or state once they are stopped or restarted. This characteristic poses a significant challenge for virtual desktop environments, where users expect their data, settings, and application states to persist across sessions.

Maintaining container images with the latest security patches, application versions, and dependency updates necessitates implementing a robust Continuous Integration/Continuous Deployment (CI/CD) pipeline. This pipeline automates building, testing, and deploying new container images, ensuring the environment remains up-to-date and secure. Through automation, the pipeline systematically applies updates and patches, reducing the risk of security vulnerabilities and compatibility issues.

Ensuring that user data and system configurations are regularly backed up is critical for effective disaster recovery. In a containerized environment, this involves a multifaceted approach, including backing up persistent storage volumes where user data is stored and ensuring container images and configuration files are included in the backup process. These backups capture the state of the entire system, enabling quick restoration in case of failure. Regular backups mitigate the risk of data loss due to hardware failures, software bugs, or security breaches.

Many applications used in virtual desktop environments have specific software requirements that might not be fully supported within containers, including dependencies on particular versions of libraries, unique system configurations, or direct access to hardware interfaces. Ensuring compatibility can involve substantial effort, requiring tweaking of container images to include necessary libraries and dependencies, modifying application code, and employing compatibility layers or emulation environments. These adjustments can introduce potential instability and inefficiencies.

Some enterprise applications are proprietary and may not support containerization out of the box, often coming with specific licensing or activation mechanisms challenging to manage within containers. Addressing these licensing challenges requires additional workarounds, such as implementing custom licensing servers to manage licenses centrally or modifying the software to function correctly within a containerized environment.

5. 2 Limitations

Moving on to the next phase, our plan includes the development phase. This phase includes the identification and creation of additional key modules, such as Process Automation Development, Web-Based Cloud Storage, Operational Dashboard, and Cloud Security. Specifically, the Process Automation Development module incorporates an Application Programmable Interface (API), enabling the web portal application to interface with a Python script application. This integration facilitates the automation of virtual machine (VM) provisioning in response to new service requests from end-users, particularly students.

Conversely, the monitoring module accommodates various applications designed for the supervision and management of the cloud platform. Simultaneously, the operational dashboard module serves as a centralized interface for configuring and controlling both physical server machines and VMs, streamlining overall operational control from a singular dashboard.

Acknowledgement

The authors would like to thank the College of Computing, Informatics and Mathematics, Universiti Teknologi MARA (UiTM), Shah Alam, Selangor, Malaysia, for the research support.

Funding

The author(s) received no specific funding for this work.

Author Contribution

Author 1 and Author 2 collaborated on crafting the literature review and supervising the article writing process. For the research methodology and fieldwork, Author 1, Author 2, and Author 3 collectively contributed. The analysis and interpretation of results were undertaken by Author 1 and Author 2.

Conflict of Interest

The authors have no conflicts of interest to declare.

References

Affouneh, S., Khlaif, Z. N., Burgos, D., & Salha, S. (2021). Virtualization of higher education during COVID-19: A successful case study in Palestine. *Sustainability*, 13(12), 6583.

Affouneh, S., Salha, S., & Khlaif, Z. N. (2020). Designing quality e-learning environments for emergency remote teaching in coronavirus crisis. *Interdisciplinary Journal of Virtual Learning in Medical Sciences*, 11(2), 135-137.

Ahmadi, M. R. (2013). Performance Evaluation of Virtualization Techniques for Control and Access of Storage Systems in Data Center Applications. *Journal of Electrical Engineering*, Vol. 64, No. 5, 2013, pp. 272–282

Ahmed, H. et. al. (2023), Exploring Performance Degradation in Virtual Machines Sharing a Cloud Server. *Journal Applied Sciences*, Vol. 13, No. 9224

Ameen, A.O, Alarape, M.A & Adewole, K.S (2019), Students' Academic Performance And Dropout Predictions: A Review. *Malaysian Journal of Computing* (*MJOC*), Vol. 4(2), pp. 278-303.

Auliya, S. et. al, (2024), Analysis and Prediction of Virtual Machine Boot Time On Virtualized Computing Environments. *Journal of Cloud Computing*, Vol. 13, No. 80

Bahrami, M., Farahbakhsh, M., Haghighat, A. T., & Gholipour, M. (2021). Virtualization and live migration: issues and solutions. *J. Comput. Based Parallel Program*, 6, 10-15.

Brown, J., Folk, K., & Swerdlow, J. (2021). The Virtualization of Schooling During the COVID-19 Pandemic. *Proceedings of the New York State Communication Association*, 2020(1), 5.

Brusakova, I. A. (2021, May). Measurement Virtualization Technologies for Intelligent Information and Measurement Systems. In *2021 XXIV International Conference on Soft Computing and Measurements (SCM)* (pp. 197-199). IEEE

Casini, D., Biondi, A., Cicero, G., & Buttazzo, G. (2021, May). Latency analysis of I/O virtualization techniques in hypervisor-based real-time systems. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS*) (pp. 306-319). IEEE.

Chen, T., & Liu, H. (2021, May). Research and Practice of Online and Offline Hybrid Teaching of Virtualization Technology Course in Higher Vocational Colleges. In *6th International Conference on Education Reform and Modern Management (ERMM 2021)* (pp. 80-83). Atlantis Press.

Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G., & Vakali, A. (2009). Cloud computing: Distributed internet computing for IT and scientific research. *IEEE Internet computing*, 13(5), 10-13.

Dong, H., Kinfe, A. T., Yu, J., Liu, Q., Kilper, D., Williams, R. D., & Veeraraghavan, M. (2021). Towards Enabling Residential Virtual-Desktop Computing. *IEEE Transactions on Cloud Computing*.

Gao, Z. (2021, September). Research on Cloud Computing Data Center Management and Resource Virtualization Technology. In *2021 4th International Conference on Information Systems and Computer Aided Education* (pp. 2067-2070).

Goel, G., Tanwar, P., Bansal, V., & Sharma, S. (2021, June). The challenges and issues with virtualization in cloud computing. In *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)* (pp. 1334-1338). IEEE.

Korikawa, T., & Oki, E. (2022). Memory Network Architecture for Packet Processing in Functions Virtualization. *IEEE Transactions on Network and Service Management*.

Kumar, R., Yadav, A. K., & Verma, H. N. (2021). An Analysis of Approaches for Desktop Virtualization and Challenges., pp. 104 -109, IEEE.

Kuo, H. C., Chen, J., Mohan, S., & Xu, T. (2020). Set the configuration for the heart of the os: On the practicality of operating system kernel debloating. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(1), 1-27.

Azzedin, F. Shawahna, A., Sajjad, F. & Abdulrahman, A.S, (2016). Performance Evaluation of VDI Environment. In *Proceeding the Sixth International Conference on Innovative Computing Technology (INTECH 2016)*,

Rashid, A., & Chaturvedi, A. (2019). Cloud computing characteristics and services: a brief review. *International Journal of Computer Sciences and Engineering*, 7(2), 421-426.

Rodríguez Lera, F. J., Fernández González, D., Martín Rico, F., Guerrero-Higueras, Á. M., & Conde, M. Á. (2021). Measuring Students Acceptance and Usability of a Cloud Virtual Desktop Solution for a Programming Course. *Applied Sciences*, 11(15), 7157.

Srivastava, P., & Khan, R. (2018). A review paper on cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, 8(6), 17-20.

Surya, P., Pachauri, P., Pachauri, A., Chaturvedi, P., Yadav, S. A., & Singh, D. (2021, November). Virtualization Risks and associated Issues in Cloud Environment. In *2021 International Conference on Technological Advancements and Innovations (ICTAI)* (pp. 521-525). IEEE.

Wan, F., Chang, N., & Zhou, J. (2020, September). Design Ideas of Mobile Internet Desktop System Based on Virtualization Technology in Cloud Computing. *International Conference on Advance in Ambient Computing and Intelligence (ICAACI) 2020*, (pp. 193-196). IEEE.

Wazan, A. S., Kuhail, M. A., Hayawi, K., & Venant, R. (2021, April). Which Virtualization Technology is Right for My Online IT Educational Labs?. In *2021 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1254-1261). IEEE.

Xiong, N., Zhou, S., Wu, Z., & Zhang, Z. (2021). Design and Research of Hybrid Cloud Desktop Scheme in Colleges and Universities. In *MATEC Web of Conferences (Vol. 336, p. 05004)*. EDP Sciences.