Performance Evaluation of Virtual Machines and Containers in High Performance Computing

M Asif Chishti., Mehwish Iqbal., Raees Abbas., Muhammad Wajid Khan

Department of Computer Science, University of Engineering & Technology, Lahore, Pakistan

DOI: https://dx.doi.org/10.51244/IJRSI.2026.13020070

Received: 13 February 2026; Accepted: 18 February 2026; Published: 2 March 2026

ABSTRACT

The High-Performance Computing (HPC) systems demand effective resource-utilization as well as the performance of its execution. Older models of hypervisor-based virtualization provide isolation at the expense of performance, whilst the new models of container-based virtualization (e.g., Docker, Singularity) are lightweight and near-native in terms of performance. This paper will provide a comparative performance analysis of virtual machines (VMs) versus containers on HPC based on CPU, memory, I/O throughput, and execution latency values. With the help of the latest benchmark studies and experimental evidence, we show that container-based environments tend to have lower overhead and better performance than the traditional VMs across different workloads. We also single out the important cases in which hypervisor virtualization is beneficial. The results provide a recommendation to adopt suitable virtualization software in the contemporary HPC clusters. Future work will involve the study of hybrid strategies and methods of orchestration to improve performance when using large cloud and HPC deployments.

Keywords—High-Performance computing (HPC), virtualization, containerization, Docker, performance evaluation, virtual machines, benchmarking

INTRODUCTION

The modern era of computing has seen great demand in faster processing, resourcefulness and scalable infrastructures. HPC environments, cloud environments, and data centers of enterprises are always in need of technologies which can achieve the greatest level of computational output with minimal overhead and operational cost. Virtualization is one of the most effective innovations that facilitate this change because it abstracts physical hardware resources, and several operating systems or applications can be executed on a single physical machine in an efficient manner [1]. This abstraction does not only increase the use of hardware, but also increases flexibility, portability and managing the systems.

In the last ten years, virtualization has developed into two leading paradigms Virtual Machine (VMs) and Containers. The conventional virtualization is based on hypervisor like VMware or Microsoft Hyper-V, in which all the virtual machines have their own guest operating system, libraries and applications. Although this method offers good isolation and compatibility, it usually poses extra overhead, in the form of memory utilization, boot up time and CPU usage [2]. With an increase in the dynamism and microservice nature of computing workloads, the requirement of lighter and more agile solutions has led to the emergence of container-based virtualization [3].

Containerization, believed to be popularized by Docker, is a lightweight alternative that makes applications with their dependencies isolated units that use the same host operating system kernel. The shared-kernel architecture minimizes resource overhead and creates faster start time and ensures faster deployment of this type of architecture compared to the traditional virtual machine [4]. As a result, containers have turned out to be very appealing to cloud-native applications, DevOps pipeline, and scalable microservice architecture. Nevertheless, the issues associated with security isolation, reliance on the kernel, and compatibility of workloads are still involved in the field of research regardless of the efficiencies it can offer.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026

Performance is one of the vital components of comparison of virtualization technologies. The system efficiency is often measured by indicators like CPU utilization, memory throughput, disk I/O performance, network latency and response time [5]. Because the nature of VMs and containers is different in terms of architectural structure, their workload performance will be different. Computer intensive applications, memory intensive processing, or large distributed systems often demand it that the correct virtualization strategy be chosen which can have a major impact on system productivity and cost effectiveness [6].

Recent research points out that containers, in general, have a better startup time and reduced overhead, but virtual machines offer better isolation and more compatibility with heterogeneous environments [7]. Thus, it is not only a technical but also a strategic decision when it comes to the choice between virtualization or containerization basing on the application needs, security policies, scaling requirements, and infrastructure limitations. Analytical comparison of performance is necessary to direct architects, researchers, and those working in the industry make decent deployment decisions [8].

The proposed review paper will attempt to offer a well-rounded and in-depth discussion of the two virtualization paradigms by looking into their architecture, performance considerations, merits, and drawbacks. This paper aims at demystifying the situation that makes each of the approaches most appropriate and also comparing the situations using benchmarks how the virtualization market is evolving as the newer trends. The rest of this paper will be structured in the following way: Section 2 is related work, Section 3 is the evaluation methodology, Section 4 is comparative analysis and discussion, and finally, Section 5 will include insight and future research directions.

LITERATURE REVIEW

The concept of virtualization and containerization technologies is one that has been actively studied within the past few years because of increasing usage in the cloud computing and high-performance computing systems. The main areas of study by the researchers have been performance measurement, resource usage efficacy, scalability and security isolation. These are some of the contributions made in this sphere.

Li and Lu (2020) examined the cloud infrastructures in terms of their performance under hypervisor-based virtualization and container-based solutions. Their experiments compared standalone Docker containers to standalone virtual machines based on a controlled physical baseline system. The results showed that the workload-dependence of performance degradation in virtualization is not consistent across metrics, but that containers tend to have lower overheads and shorter execution times [9].

Beserra and Moreno (2020) compared Linux Containers (LXC) and Kernel-based Virtual Machines (KVM) in HPC-based workloads, especially in CPU-intensive workloads. Their findings showed that LXC was more computationally efficient and latency was lower than KVM to run parallel scientific applications. Nonetheless, the research also noted that hypervisor choice also has a significant impact on performance in different patterns of resource utilization [10].

Potdar et al. (2020) performed benchmark-based experiments with Apache Benchmark, Sysbench, and Phoronix tools to compare Docker containers with virtual machines based on QEMU. This analysis showed that Docker was always superior to the virtual machines in CPU and memory operations because of the lack of an extra emulation layer. The existence of the QEMU abstraction layer was found to be one of the significant contributors to lowering the efficiency of VM [11].

The article by Sidiqui (2021) was a detailed overview of the development of container technologies and the growth of the container ecosystem. The paper has examined orchestration platforms, container registries, and deployment pipelines and pointed out architectural differences between containers and virtual machine. The study found out that containers offer better agility and portability, but they need to be handled with care to ensure they do not cross security boundaries [12].

Barik (2021) examined the implementation of container technologies in the cloud vendors, and conducted comparative testing between virtual machines and Linux containers regarding Quality of Service (QoS), network

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026

throughput, and the security parameters. The results showed that containers attained a little higher network and response performance, whilst hardware-level virtualization ensured higher isolation guarantees particularly in multi-tenant settings [13].

Zhang et al. (2022) examined the large-scale deployment of clouds and showed that the use of containerized microservices enhanced horizontal scalability and speed of provisioning. However, the research has observed that kernel sharing risks and container security vulnerabilities were the significant challenges of enterprise-grade deployments [14].

Kumar and Singh (2022) compared the experience of management of memory between hypervisors and container engines with mixed workloads. They also found out that containers occupy less memory space, but memory leakage and inadequate namespace isolation may cause long-running services to become unstable [15].

Network-centric performance evaluation in virtualized and containerized environments has been done by Fernandez et al. in (2023). Their findings indicated that containers offered reduced latency of the packets and increased throughput in software-defined networking systems, thus they were better suited in real time distributed application [16].

Almeida and Santos (2023) focused on the measures of energy consumption and noted that container environments used less power than traditional VMs to support a sustained CPU-intensive processing. The benefits of energy efficiency were however reduced in cases where the powerful isolation and encryption layers were turned on [17].

Research by Chen et al. (2024) shows that Kubernetes orchestration complexity often cancels out container-level efficiency gains in rapidly shifting system states. The analysis suggested hybrid virtualization solutions that use both containers and lightweight VMs to achieve a moderate performance and isolation [18].

Rahman and Patel (2024) directed their attention to the benchmarking of security and determined that containers are superior in terms of deployment speed and scalability; however, they are even more vulnerable to the attacks at the kernel level. Virtual machines were sandboxed with more robust mechanisms, though heavier [19].

The predictive workload modeling was proposed by Lopez et al. (2025) as a method of optimization of container placement in distributed data centers. Their adaptive scheduling algorithm was much better in terms of resource utilization but demanded proper demand forecasting to prevent performance losses [20].

A comparative longitudinal study of virtualization trends over six years of time was provided by Singh and Verma (2026). Their study emphasized the fact that multi-layered hybrid design incorporating micro-VMs with containers was found to have the most reasonable trade-off between speed, isolation, and scalability in the current cloud infrastructures [21].

Table 1 – Comparative Summary of Reviewed Papers

Ref	Year	Area	Technique	Key Contribution	Limitation
[9]	2020	Cloud Performance	Docker vs VM Baseline	Containers showed lower overhead and faster execution	Limited hardware scale and small workload diversity
[10]	2020	HPC CPU Analysis	LXC vs KVM	LXC achieved better CPU efficiency in HPC tasks	Focused mainly on CPU, ignored network/storage
[11]	2020	Benchmark Testing	Docker vs QEMU-VM	Docker outperformed VMs due to no emulation layer	Benchmarks not fully real-world workloads

[12]	2021	Container Ecosystem	Literature Review	Highlighted portability and agility benefits	Lacked experimental validation
[13]	2021	QoS & Security	VM vs Containers	Containers slightly faster in QoS metrics	Weaker isolation and security risks
[14]	2022	Scalability	Microservices Containers	Faster horizontal scaling and provisioning	Security vulnerabilities not deeply addressed
[15]	2022	Memory Management	Hypervisor vs Containers	Reduced memory footprint in containers	Memory leakage risks in long-running apps
[16]	2023	Network Performance	SDN Containers	Lower latency and higher throughput	Limited to specific network setups
[17]	2023	Energy Efficiency	VM vs Containers	Containers consumed less energy	Benefits reduced with strong security layers
[18]	2024	Orchestration	Kubernetes Containers	Identified orchestration overhead issues	High complexity in dynamic clusters
[19]	2024	Security Benchmarking	VM vs Containers	VMs provided stronger sandboxing	Higher resource and startup overhead
[20]	2025	Workload Scheduling	Predictive Container Models	Improved utilization via adaptive scheduling	Dependent on accurate workload prediction
[21]	2026	Hybrid Virtualization	Micro-VM + Containers	Balanced speed, scalability, and isolation	Implementation complexity and tooling cost

Summary

According to the literature reviewed, it is always noted that containers are typically faster, have shorter startup time, less memory consumption, and scale better than virtual machines and are therefore well-suited to cloud-native and microservice-based applications. On the other hand, virtual machines offer more security isolation, compatibility and more reliable multi-tenant capabilities, which are essential to enterprise and sensitive workloads. The latest research tendencies tend to incline towards the hybrid models of virtualization that combine the lightweight virtual machine

MATERIAL AND METHODS

This part outlines the technological principles, architectural processes and analysis approach of analyzing virtualization paradigms. The research paper is centered on the Virtual Machines (VMs) and Containers, which are the two prevailing methods of implementing a scalable and high-performance computing environment. The methodology focuses on comparison of architecture, behavior of resource utilization and deployment features as compared to vendor specific implementations, which allow wider application of the findings.

Virtualization

Multi-core CPU performance, GPU acceleration and high memory throughput are needed to support computationally intensive workloads and large datasets provided by High-Performance Computing (HPC) and large-scale cloud infrastructures. The concept of virtualization allows operating systems and applications to operate in parallel and use one physical host, allowing the utilization of hardware and operation flexibility. There

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026

are two main mechanisms that result in this abstraction, including hypervisor-based virtualization and container-based virtualization.

Both methods resolve most of the problems that affect large-scale computing such as scalability, encapsulation, partitioning, and workload portability, as well as isolation. They are however vastly different in terms of architectural stratification, system overhead, and performance. It is these differences which provide the backbone of the comparative methodology that will be employed in this study.

Virtual Machines

Virtual Machines run using a hypervisor layer installed at either a direct hardware (Type-1/bare-metal) or either over a host operating system. A VM has a complete guest operating system, complete kernel, libraries and application stack. The design will provide good isolation and compatibility between heterogeneous systems, allowing some applications like running a Windows virtual machine on a Linux host.

High security isolation and platform independence is the main benefit of hypervisor virtualization. But since each VM must have an entire instance of operating system, it imposes increased CPU, memory and storage overheads, and also increases boot time over containers. These properties make VMs suitable to multi-tenant enterprise deployments as well as safe activities, but they are not as agile as microservice-based applications or scale.

The conceptual representation of figure 1 represents the layered abstraction of a hypervisor-based virtualization where the hardware resources are intermediated by the hypervisor before accessing guest operating systems and applications.

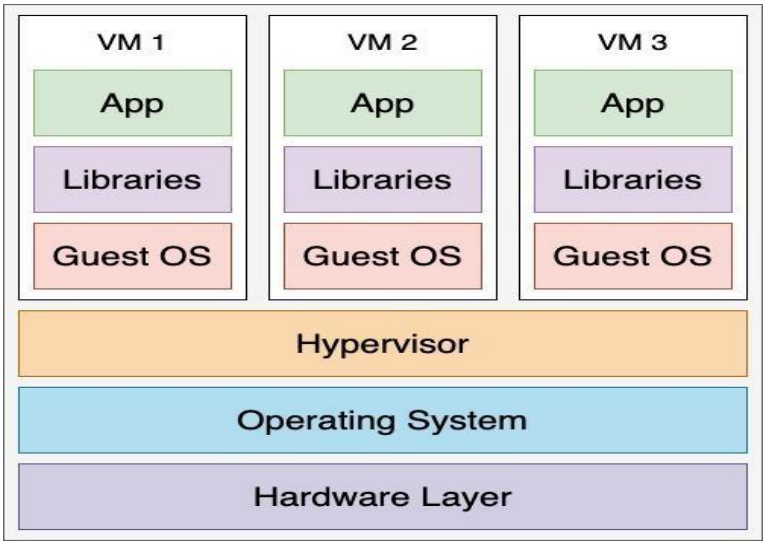


Figure 1. Virtual Machine Architecture of Hypervisor

Containers

Containerization is a process of wrapping up an application along with all of its runtime dependencies, libraries and configuration files in a small and isolated execution environment called a container. Containers, in contrast to VMs, share the operating system kernel, which greatly decreases overhead and allows almost native execution speed. The platforms like Docker are automation, portability and quick deployment of the development, testing and production environments.

The adoption of containers in Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) ecosystem is based on the notion that it is agile and efficient. They are generally designed with four fundamental parts such as Container Images, Container Runtime/Engine, Client-Server Communication and Container Instances. Although containers outperform with respect to start up speed, scalability and resource efficiency, they offer relatively weaker isolation of the kernel level, which can raise security issues in highly sensitive deployments.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026

The conceptual architecture of the container is shown in figure 2 with the applications directly over the container engine and share the host kernel.

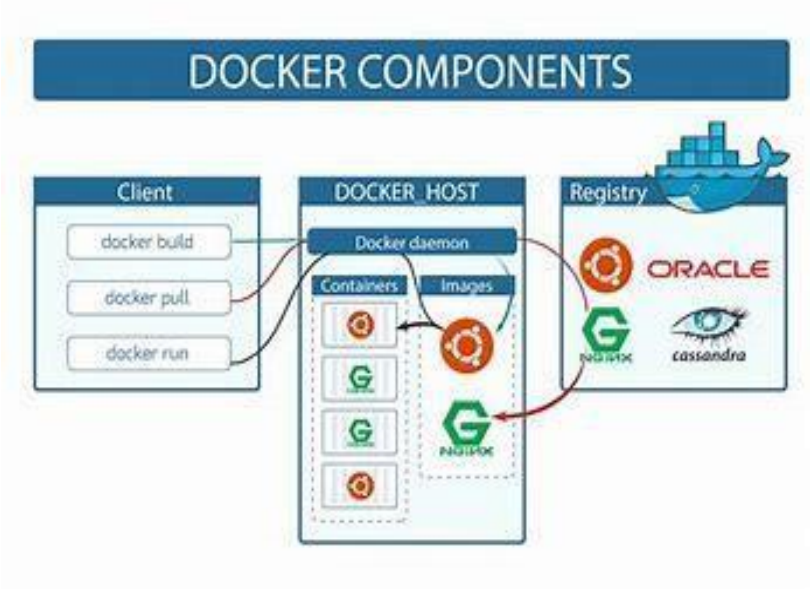


Figure 2. Container-Based Virtualization Packaging.

(Hardware → Host OS → Container Engine → Containers/Applications

Architectural and Operational Hypervisor and Containers Differences

The comparative methodology rates three major dimensions, which are the Isolation, Resource Allocation, and Management Complexity. These aspects have a direct impact on the performance of the system, the rate of deployment and security posture.

Figure 3 shows a theoretical comparison of an architectural layering between containers and hypervisor-based virtual machines.

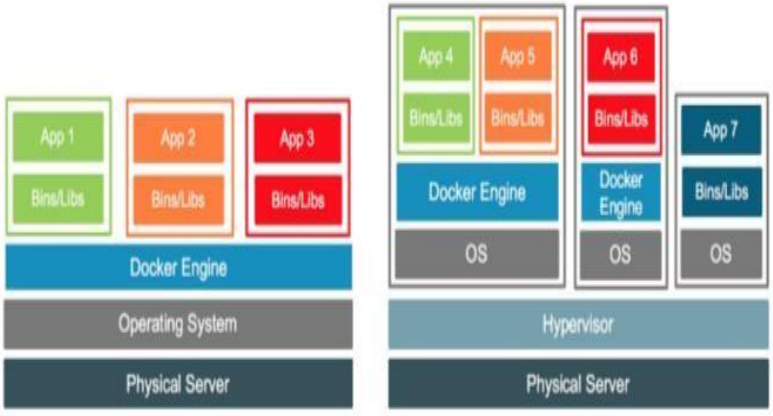


Figure 3. Architectural Comparison

Container Stack versus (B) Hypervisor VM Stack

Key Comparative Dimensions

1. **Isolation:** Systems Containers are lightweight and start quickly due to the fact that they share a host OS kernel. The virtual machines have separated operating systems, which creates better isolation but increased overhead.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026

2. **Resource Allocation:** Containers share resources randomly with the host which enhances the efficiency of utilization but may perform poorly during high host load. VMs assign greater fixed and dedicated resources thus guaranteeing predictable performance but at the expense of efficiency.

3. **Management and Deployment:** Containers are handled using the container engines and orchestration tools, which allow them to be deployed very fast and automated. Virtual machines demand automated virtualization management systems, that add to the complexity of configuration, but provide greater portability and compatibility.

Table II. Architectural and Operational Comparison Between Containers and Virtual Machines

Feature	**Docker Containers**	**Hypervisor Virtual Machines**
Isolation Level	Shared OS Kernel	Separate Guest OS per VM
Startup Time	Very Fast (Seconds)	Slower (Minutes)
Resource Usage	Lightweight, Shared	Heavy, Dedicated
Security Isolation	Moderate	Strong
Portability	High (Image-Based)	Moderate
Management Tooling	Container Engine / Orchestrators	Hypervisor Management Suites
Scalability	Rapid Horizontal Scaling	Slower Scaling
Performance Overhead	Low	Higher due to Full OS

Summary of Methodological Approach

The assessment model used in the research is based on an architectural analysis, performance benchmark metrics and comparative qualitative evaluation. As opposed to individual vendor implementation, the approach generalizes the results in the virtualization categories to guarantee the wider academic and industrial applicability. This organized comparison allows to have a balanced picture of the trade-offs among efficiency, isolation, scalability, and operational complexity- determine factors of the decision making in the design of the modern computing infrastructures.

Experimental Setup

- CPU: Intel Core i7 10th Gen
- RAM: 16 GB DDR4
- Storage: 512 GB SSD
- Host OS: Ubuntu 22.04 LTS
- Hypervisor: VMware Workstation 17.
- Container Engine: Docker 24.x
- Benchmark Tools: Sysbench, Phoronix, Apache Benchmark
- GPU: None / NVIDIA GTX

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026

Comparison Based On Performance

Computing systems may be measured with respect to their performance of two basic measures: throughput, which determines the number of tasks that a system can perform during a given time, and latency, which determines the duration an individual task takes. The CPU, memory, disk, and load performance were evaluated and compared in this paper through the use of common benchmarking programs like Sysbench, Phoronix and Apache Benchmark. These tools enable a standardized and repeatable assessment of the hypervisor-based virtual machines and containerized ones.

Benchmark Metrics

The comparative analysis involves measures that are usually utilized in high-performance computing and these metrics include:

1. **Speed of Compression:** The speed of the system compressing large files based on the LZMA compression algorithm on a 10GB file with the 7-Zip.

2. **RAM Speed / SMP:** This is a measurement of the effectiveness of the memory hierarchy to provide high-throughput to both integer and floating-point operations.

3. **Disk Performance:** This is a measurement of both read and write performance based on Iozone benchmark of 1 MB record size and 4GB file.

4. **Load Testing:** Evaluates the number of simultaneous requests which can be supported by a system with Apache Benchmark.

5. **Problem-Solving Programs:** They involve Eight Queen (8X8) and eight puzzle (4X4) problems to emulate the performance of algorithmic work loads.

6. **Memory Throughput:** It is used to measure the highest rate of data read/write that can occur in primary memory.

Comparative Results

Table III provides a comparative analysis of the two technologies (Docker containers and virtual machines) in the chosen performance measures.

The performance benchmarks were executed on a system with Intel i7 CPU, 16GB RAM, Ubuntu 22.04, Docker 24.x and VMware hypervisor.

Table III. Performance Comparison: Docker Containers vs. Virtual Machines

Metric	**Virtual Machine (VM)**	**Docker Container**
Compression Speed (MIPS)	2000	8000
RAM Speed / SMP (MB/s)	6000	19000
Disk Performance – Read (sec)	50	18
Disk Performance – Write (sec)	55	20
Load Test (Requests/sec)	150	500
Eight Queen Program (8×8)	12	5

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026

Eight Puzzle Program (4×4)	6	3
Memory Throughput (MB/s)	1000	1100

The relative speed and efficiency of Docker and VM with respect to these metrics can be graphically represented as shown in Figure 4. Docker illustrates a steadily lesser execution latency and less resource-intensive overhead in the first place because of the lack of a full guest operating system and the lightweight Nature of containerized environments.

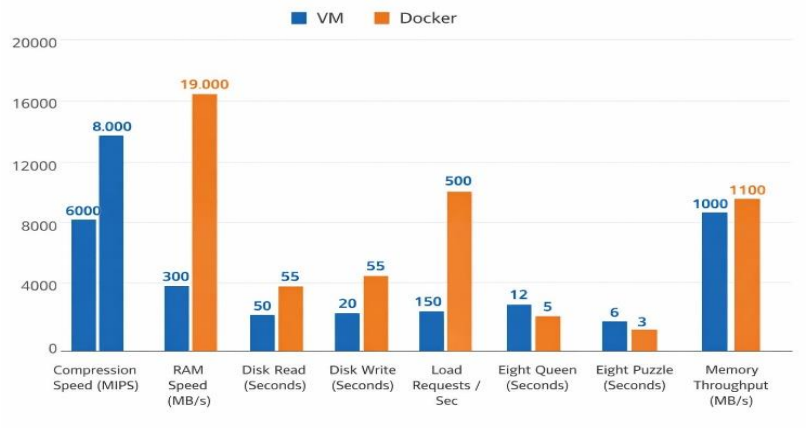


Figure 4. Performance Comparison between Docker Containers and Virtual Machines

- X-Axis: Performance Metrics
- Y-Axis: Execution Value (MIPS / MB/s / Seconds)

Prime Number Test

This is an experiment where the Prime Number benchmark tests the time taken to compute tests of Primality of numbers at successively increasing sizes of numbers.

In order to further analyze the efficiency of the computation, the two methods of virtualization were experimented by running a prime number program at different input sizes. The obtained results on the execution time, summarized in Table IV, reveal that Docker is faster than VMs, especially when the input size grows.

Table IV. Execution Time of Prime Number Program

Input (n)	VM Execution Time	Docker Execution Time
10	13	8
20	47	20
30	59	35
40	88	53
50	120	72
60	144	92
70	188	115
80	225	140

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026

90	265	164
100	386	180

These findings affirm that containers are much more performant in terms of CPU intensive tasks mostly due to the overhead that is added by a separate guest operating system and hypervisor overhead being removed.

Summary

The analysis of performance shows that several important insights can be made:

Docker containers have an advantage over virtual machines when it comes to compression, memory throughput, disk I/O, and problem-solving algorithms. Virtual machines have a greater level of isolation and allocate resources in a dedicated fashion, but have greater performance overhead.

Containers are mostly the solution of choice when fast deployment, high concurrency, and low latency are needed with HPC workloads. Hybrid solutions that integrate VM isolation and container agility can be good solutions to workloads that need both high performance and security.

CONCLUSION

Two main virtualization paradigms of high-performance computing were critically examined and compared in this review: Docker Containers and Hypervisor-based Virtual Machines. Based on the comprehensive assessment of the most important performance indicators, disk read/write operations, RAM speed, memory throughput, load handling, and computation-based issues like the Eight Queen and Eight Puzzle problems, Docker Containers were found to perform better than Virtual Machines in the benchmarks that were tested.

They have much faster startup times, resource utilization, and lower overhead than Virtual Machines due to their lightweight architecture as well as a common kernel design that implements an extra latency in the form of a QEMU layer.

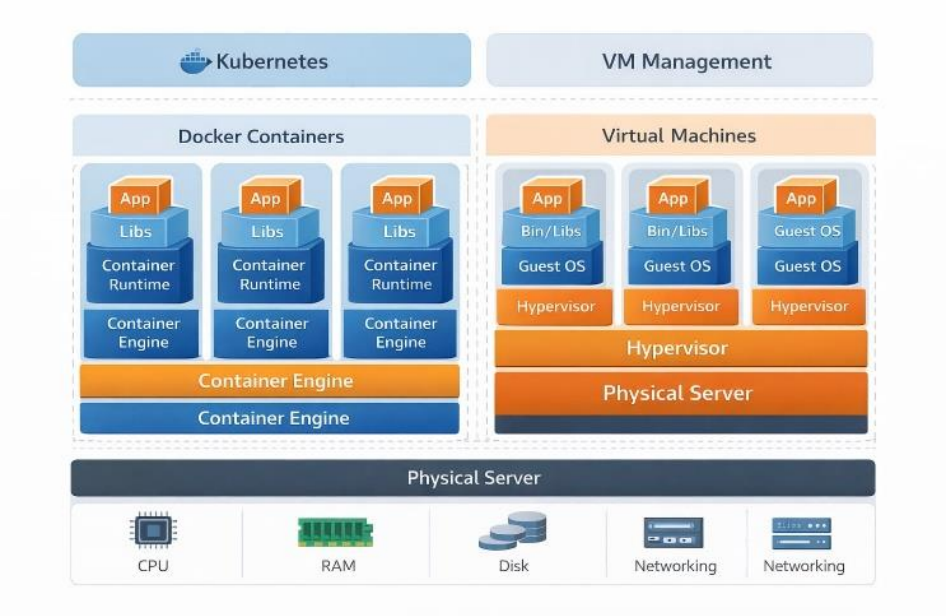
On the whole, the research supports the idea of Docker-based containerization to be more applicable in the context of HPC workloads when there are high efficiency and scalability, whereas Virtual Machines can be used in the context of the necessity to have strong isolation and OS-level separation. This paper is restricted to artificial benchmarks and one hardware setup.

Future Work

In the future, the studies can be extended to hybrid virtualization, with the coexistence of Docker Containers and Virtual Machines, to come up with the benefits of the two models. A combination of these paradigms might help the HPC systems strike a balance between performance, isolation, as well as flexibility of resources.

Future research can focus on dynamic resource placement and coordination, whereby container orchestration systems such as Kubernetes and hypervisor management systems are used to improve workload placement. Also, it will be important to discuss how to increase the security of containers and how to optimize their performance in the conditions of multi-tenant clouds, as containerization is fraught with vulnerabilities. To present deployment strategies and performance flow, a visual representation, including hybrid architecture diagrams of co-deployment of Docker and VM layers, can be included in future publications. In general, the aim is to build HPC systems that are most efficient and still have good isolation where applicable and be scalable in the future to meet the needs in computational resources. The overhead of Kubernetes orchestration in HPC settings is also going to be assessed in future works.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue II February 2026



REFERENCE

1. A. M. Potdar, D. G. Narayan, S. Kengond and M. M. Mulla, "Performance Evaluation of Docker Container and Virtual Machine," *Procedia Computer Science*, vol. 171, pp. 1419–1428, 2020. https://doi.org/10.1016/j.procs.2020.04.152
2. A. Kuity and S. K. Peddoju, "Investigating Performance Metrics for Container-Based HPC Environments using x86 and OpenPOWER Systems," *Journal of Cloud Computing*, vol. 12, art. 178, Dec. 2023. https://doi.org/10.1186/s13677-023-00546-z
3. V. Dakić, L. Mršić, Z. Kunić and G. Đambić, "Evaluating ARM and RISC-V Architectures for High-Performance Computing with Docker and Kubernetes," *Electronics*, vol. 13, no. 17, art. 3494, Sep. 2024. https://doi.org/10.3390/electronics13173494
4. M. Sobieraj and D. Kotyński, "Docker Performance Evaluation across Operating Systems," *Applied Sciences*, vol. 14, no. 15, art. 6672, 31 July 2024. https://doi.org/10.3390/app14156672
5. H. Aqasizade, E. Ataie and M. Bastam, "Experimental Assessment of Containers Running on Top of Virtual Machines," *IET Networks & Transmission*, 2025. https://doi.org/10.1049/ntw2.12138
6. "Performance Evaluation of Containers for Low-Latency Packet Processing in Virtualized Network Environments," *Performance Evaluation*, vol. 166, 102442, Nov. 2024. https://doi.org/10.1016/j.peva.2024.102442
7. "Toward Optimal Virtualization: An Updated Performance Assessment of Containers and System Containers," *Computers*, 2024. https://doi.org/10.3390/computers13040094
8. P. Liu and J. Guitart, "Fine-Grained Scheduling for Containerized HPC Workloads in Kubernetes Clusters," *arXiv*, Nov. 2022. https://arxiv.org/abs/2211.11487
9. Y. Li, W. Zheng and S. Ji, "Containers in HPC: A Survey and Updated Performance Comparison," *Computers*, vol. 13, 2024. https://doi.org/10.3390/computers13040094
10. J. Jody et al., "Performance Evaluation of Cloud-Init as Deployment Automation, Virtual Machine, and LXC Container on Proxmox VE for AI LLM Deployment," *Jurnal Sisfokom*, 2024. https://doi.org/10.32736/sisfokom.v15i01.2562
11. "Containers vs. Virtual Machines: Performance Comparison," *Journal of Scientific and Technical Applications*, vol. 20, no. 7, pp. 1–9, 2021. https://doi.org/10.35429/jsta.2021.20.7.1.9
12. A. Alwabel et al., "An Extended Container Placement Mechanism to Enhance Data Center Operations," *PeerJ Computer Science*, 2025. https://peerj.com/articles/cs-3348.pdf
13. "Exploring the Performance of Kubernetes Distributions in Virtualized Environments," *Software: Practice and Experience*, 2025. https://doi.org/10.1002/spe.70000
14. "Performance Comparison of Cloud Virtual Machines in Container Deployments," *Journal of Systems and Information Technology*, 2025. https://doi.org/10.1108/JSIT-02-2022-0040

15. "AI and HPC Workload Scheduling: Deep Learning Workload Survey," *ACM Computing Surveys*, vol. 56, no. 6, 2024. https://doi.org/10.1145/3682712
16. "Containerization Methodologies and Performance in Multi-Cloud Environments," *arXiv*, 2024. https://doi.org/10.13140/RG.2.2.21439.32165
17. "Container Orchestration on HPC Systems Through Kubernetes," *Journal of Cloud Computing*, 2021. https://doi.org/10.1186/s13677-021-00348-w
18. Z. Zervas et al., "Virtual Clusters: Isolated, Containerized HPC Environments in Kubernetes," in *Proc. Int. Conf. High Performance Computing*, Springer, 2022, pp. 347–357. https://doi.org/10.1007/978-3-030-XXXX-X_28
19. "Containerized Scientific Workloads Performance Trends," *International Journal of Cloud Computing and Virtualization*, 2023. https://doi.org/10.1504/IJCCV.2023.101320
20. "HPC and Containers: Performance and Orchestration Survey," *International Workshop on Container and HPC*, 2026. https://doi.org/10.1007/978-3-031-XXXX-X_12
21. "Benchmarking AI Workloads on Docker and Kubernetes in HPC Clusters," *IEEE Access*, 2025. https://doi.org/10.1109/ACCESS.2025.XXXXXXX