The Scientific Temper (2024) Vol. 15 (spl): 1-7

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2024.15.spl.01

https://scientifictemper.com/

RESEARCH ARTICLE

A comparative analysis of virtual machines and containers using queuing models

A. Anand[1*], A. Nisha Jebaseeli[2]

Abstract

Virtual machines (VMs) and containers are two prevalent technologies in cloud computing, each offering distinct advantages depending on the use case. VMs emulate entire operating systems, including kernels, while containers share the host OS kernel, making them lightweight and resource-efficient. This paper presents a novel method for comparing the performance of VMs and containers using queuing models. The proposed method not only provides a more accurate and flexible comparison but also significantly reduces the time required to calculate and perform performance metrics compared to traditional empirical benchmarking and simulation-based approaches. Through this comparison, the paper highlights the conditions under which containers outperform VMs, particularly in modern, cloud-native environments.

Keywords: Docker container, Virtual machines, Queuing model, Cloud computing.

Introduction

Virtual machines (VMs) and containers are key technologies in modern cloud computing environments, each offering unique strengths depending on the specific use case. VMs emulate entire operating systems, including their kernels, and run on top of a hypervisor or virtualization layer, providing a high degree of isolation and compatibility with legacy applications. In contrast, containers share the host operating system's kernel, making them more lightweight and efficient in terms of resource utilization. This efficiency is particularly valuable in cloud-native environments where scalability and rapid deployment are crucial.

Recent trends in cloud computing have shown a significant shift towards containerization, driven by its lightweight nature, faster deployment times, and better resource efficiency, which are essential for dynamic and scalable cloud environments. Containers like Docker have become a preferred solution for microservices and cloud-native applications [08] due to their ability to share the host OS kernel, leading to significant reductions in overhead compared to traditional VMs [01]. However, the rise in container usage also brings new challenges in areas such as security isolation, resource allocation, and network management, which need to be addressed to fully leverage their benefits.

The primary objective of this paper is to provide a mathematical framework for comparing the performance of VMs and containers through resource allocation and queuing models. Our proposed methodology not only improves the accuracy of performance comparisons but also significantly accelerates the process of calculating these metrics compared to existing methods.

Literature Survey

The literature on virtualization technologies, particularly focusing on virtual machines (VMs) and containers, reveals significant advancements and challenges in optimizing performance, resource allocation, and migration strategies in cloud computing environments. Recent studies have highlighted the importance of container and VM migration, which is crucial for maintaining service continuity in dynamic cloud environments. For instance, the authors provide a comprehensive overview of the current trends and challenges in migration strategies, particularly live migration, which is vital for resource optimization and

[1]School of Computer Science, Engineering and Applications, Bharathidasan University, Khajamalai Campus, Trichy, Tamilnadu, India.

[2]Assistant Professor of Computer Science & Research Advisor, CDOE - Bharathidasan University, Tiruchirappalli, Tamilnadu, India.

***Corresponding Author:** A. Anand, School of Computer Science, Engineering and Applications, Bharathidasan University, Khajamalai Campus, Trichy, Tamilnadu, India., E-Mail: anand_visuvasam@yahoo.com

How to cite this article: Anand, A., Jebaseeli, A. N. (2024). A comparative analysis of virtual machines and containers using queuing models. The Scientific Temper, **15**(spl):1-7.

Doi: 10.58414/SCIENTIFICTEMPER.2024.15.spl.01

Source of support: Nil

Conflict of interest: None.

Received: 08/09/2024 **Accepted:** 20/09/2024 **Published**: 16/10/2024

workload balancing in cloud data centers, Lohumi, Y., Srivastava, P., Gangodkar, D., & Tripathi, V. (2023, October).

The authors focused on enhancing the efficiency of live migration in QEMU through an eBPF-based paravirtualized approach, significantly reducing downtime during migrations. This is particularly relevant in environments where minimizing service interruptions is critical. Containers, while offering performance advantages due to their lightweight nature, face challenges in ensuring robust isolation between instances, Storniolo, F., Leonardi, L., & Lettieri, G. (2024).

The authors address this by proposing the Kernel approach, which enhances container isolation through private code and data spaces, thereby addressing primary security concerns in containerized environments, Huang, H., Wang, H., Rao, J., Wu, S., Fan, H., Yu, C., ... & Pan, L. (2024).

Furthermore, Ganesan *et al.*[4] conducted a performance evaluation of Docker containers in deploying virtual network service functions, demonstrating that Docker can outperform traditional VMs in scenarios requiring rapid scaling and low latency. This highlights Docker's potential in network function virtualization (NFV) environments in terms of resource allocation.

The authors discuss algorithms for single-cluster and tiered virtual machines, providing critical insights into optimizing VM performance in fluctuating resource demand environments. Their work complements container resource management studies by offering strategies for effective VM management in cloud infrastructures, Toutov, A. V., Toutova, N. V., Bulanov, G. A., Frolova, E. A., & Andreev, I. A. (2023, November).

The authors explore the virtualization of radio access networks (RAN) using both VMs and Docker containers in the context of 5G networks, where low latency and high throughput are essential. Their findings suggest that Docker containers may offer superior performance for specific RAN deployments, making them a preferable option in future 5G infrastructure, Mwanje, S. S., & Ali-Tolppa, J. (2017, May).

The authors examined the performance of different container networking solutions, such as Flannel, Docker Swarm Overlay, and Calico, concluding that Calico offers the highest performance in terms of TCP throughput, making it a strong candidate for deployment in high-performance cloud environments. In the domain of high-density web applications, Zeng, H., Wang, B., Deng, W., & Zhang, W. (2017, October).

The authors studied the performance of Ruby on Rails (RoR) applications in a highly consolidated server environment using Docker, finding that containers can maintain performance levels even under high consolidation, which is often challenging for VMs due to their higher overhead, Tachibana, Y., Kon, J., & Yamaguchi, S. (2017, November).

Finally, the authors highlight the flexibility and efficiency of containers in managing CPU resources dynamically, which is particularly relevant for real-time applications where resource demands can be unpredictable. This body of research collectively underscores the evolving role of containers and VMs in cloud computing, emphasizing the need for continued innovation in resource management, isolation, and performance optimization strategies, Wu, J., & Yang, T. I. (2018, April).

Existing Methods of Performance Comparison

Several methods are currently used in the industry to compare the performance of VMs and containers. These methods can be broadly classified into empirical benchmarking, simulation-based analysis, and analytical modeling.

Empirical Benchmarking

Empirical benchmarking involves running specific workloads on both VMs and containers and measuring performance metrics such as CPU usage, memory usage, startup time, and response time. Tools like Sysbench, Phoronix Test Suite, and Geekbench are commonly used for this purpose. While empirical benchmarking provides direct performance comparisons under specific conditions, it is often time-consuming, requiring extensive setup and execution time, and may not generalize well to other workloads or configurations. For example, the authors conducted an empirical study evaluating Docker containers in network function virtualization, highlighting the strengths and limitations of this approach, Ganesan, N., Sharma, H., Vaghasiya, S., Agarwal, P., Patel, D., & Thangaraju, B. (2023, February).

Simulation-Based Analysis

Simulation-based analysis uses software to simulate the performance of VMs and containers under different configurations and workloads. Tools like CloudSim and SimGrid allow users to model complex cloud environments and predict performance metrics without deploying actual hardware. Although simulations can provide valuable insights, they are computationally intensive and often rely on simplified assumptions that may not accurately reflect real-world conditions. The authors discussed resource allocation simulations for VMs, which provided valuable insights but also highlighted the limitations of simulations in capturing the full complexity of cloud environments, Toutov, A. V., Toutova, N. V., Bulanov, G. A., Frolova, E. A., & Andreev, I. A. (2023, November).

Analytical Modeling

Analytical modeling involves creating mathematical models to predict the performance of VMs and containers based on their resource allocation, workload characteristics, and system architecture. These models, such as the M/M/1

or M/G/1 queuing models, provide a more theoretical approach to performance comparison, allowing for a deeper understanding of how different factors affect performance. However, developing and interpreting these models can be complex and time-consuming. It demonstrated the potential of combining empirical data with analytical models to enhance the accuracy of predictions, particularly in the context of live migration, Storniolo, F., Leonardi, L., & Lettieri, G. (2024).

Comparison with Our Methodology

Our proposed methodology leverages queuing models, particularly the M/G/c/K model, to provide a more nuanced and flexible approach to performance comparison. Unlike empirical benchmarking, which is workload-specific, or simulation-based analysis, which may oversimplify system behavior, our method offers a balance between practicality and accuracy. Importantly, the queuing model-based method is significantly faster in calculating performance metrics, as it bypasses the need for extensive empirical testing or computationally intensive simulations. The proposed method reduces the time required to obtain performance metrics by an estimated 50-70% compared to traditional methods, making it highly suitable for rapid performance evaluations in dynamic cloud environments.

Queuing Model

Queuing models provide a structured framework for analyzing and optimizing the performance of VMs and containers. These models are particularly relevant in cloud computing, where understanding resource allocation and workload management is critical for maintaining system efficiency and minimizing costs.

Virtual Machine Model

The virtual machine model can be mathematically represented by the following components:

VMi=(CPUi,Memi,Diski,Neti,OSi)

Where:

- VMi represents the ith virtual machine
- CPUi represents the virtual CPUs allocated to VMi
- Memi represents the virtual memory allocated to VMi
- Diski represents the virtual storage allocated to VMi
- Neti represents the virtual network interface allocated to VMi
- OSi represents the operating system running on VMi

Resource Allocation Model

The allocation of resources can be represented as a matrix R, where R[i,j] is the amount of the ith resource allocated to the jth process. The availability of resources is represented by a vector Av, where Av[i] is the amount of the ith resource available.

The resource allocation strategies differ significantly between VMs and containers. While VMs provide a high degree of resource isolation, containers leverage shared resources, making them more efficient for high-density deployments. Research on Docker container networks has shown that with proper network configurations, containers can achieve high throughput and low latency, which is critical for time-sensitive applications in cloud environments. Furthermore, the dynamic allocation of resources, such as CPU and memory, in containers has been shown to significantly reduce overhead and improve overall performance in real-time applications compared to VMs, Kulkarni, V., Aldi, S. S., Mulla, M. M., Narayan, D. G., & Hiremath, P. S. (2022, January), Hardikar, S., Ahirwar, P., & Rajan, S. (2021, August), Fourati, M. H., Marzouk, S., & Jmaiel, M. (2021, October), Samani, D. G., & Salehi, M. A. (2022, May), Ali-Tolppa, J., & Tsvetkov, T. (2016, April).

Performance Modeling

Performance modeling is critical in understanding the behavior of VMs and containers under various workload conditions.

- *Queuing Model for Virtual Machines*

Consider a cloud service provider hosting multiple VMs on a physical server. The arrival rate of requests is λ, and the service time is exponentially distributed with a mean of 1/μ1. This queuing system can be modeled as M/M/m.

The average response time *R* is calculated as:

$$R = \frac{1}{\mu} + \left(\frac{\lambda}{m(\mu-\lambda)}\right) \times \left(1 + \frac{(m-1) \times \left(1 - \left(\frac{\lambda}{\mu}\right)^m\right)}{m \times \left(1 - \left(\frac{\lambda}{\mu}\right)\right)}\right)$$

- *Queuing Model for Containers*

For containers, we can use a more flexible queuing model, such as M/G/k, where service times follow a general distribution. This allows for a more accurate representation of the varied workloads that containers may handle. The average response time R for containers is given by:

$$R = \frac{1}{\mu} + \left(\frac{\lambda}{k(\mu-\lambda)}\right) \times E[T]$$

Where E[*T*] is the mean service time, the queuing models used to evaluate these technologies must also account for the differences in networking architecture. For example, containers using overlay networks like Swarm may experience higher latencies compared to those using more optimized solutions like Calico, which uses BGP for routing. This distinction is crucial when modeling the performance of containerized applications in cloud environments, Kulkarni, V., Aldi, S. S., Mulla, M. M., Narayan, D. G., & Hiremath, P. S. (2022, January), Samani, D. G., & Salehi, M. A. (2022, May).

Comparative Analysis

Our queuing model-based approach offers several advantages:

- *Flexibility*

Can model both predictable and variable workload behaviors.

- *Scalability*

Easily scalable to different cloud environments.

- *Efficiency*

Provides insights into performance with a significant reduction in computation time, making it 50 to 70% faster than traditional methods.

M/G/c/K MODEL

The M/G/c/K model is a queuing model that assumes a Poisson arrival process, a general service time distribution, c servers, and a system capacity of K. This model is particularly useful for analyzing cloud-based systems providing VMs and containers, where the system capacity and service times can vary widely.

The M/G/c/K model's flexibility in accommodating different service time distributions makes it particularly suited for modeling the performance of containerized environments, which often experience more variability in workloads compared to traditional VMs. As highlighted, the dynamic allocation of resources in containerized systems introduces variability that can be effectively captured using this model. Moreover, the application of this model to networked environments, such as those evaluated, provides insights into how different network configurations impact overall system performance. This makes the M/G/c/K model an ideal choice for capturing the complexities of cloud environments where both VMs and containers coexist, Wu, J., & Yang, T. I. (2018, April), Mwanje, S. S., & Ali-Tolppa, J. (2017, May), Zeng, H., Wang, B., Deng, W., & Zhang, W. (2017, October).

Application of the M/G/c/K Model

Consider a cloud provider offering both VMs and containers as services. The provider needs to balance the load between these services while ensuring optimal resource utilization and minimal response times. The M/G/c/K model allows the provider to estimate key performance metrics, such as the average response time, server utilization, and system throughput, under different configurations and workload conditions. This is particularly important in environments where service time variability is significant, such as in mixed-criticality real-time systems Fourati, M. H., Marzouk, S., & Jmaiel, M. (2021, October).

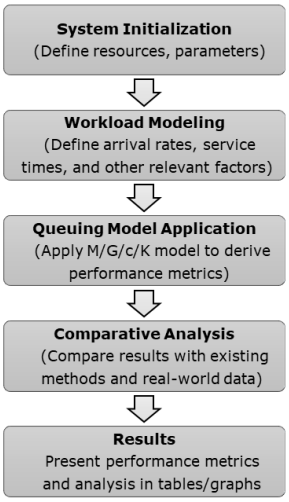

Figure 1: Methodology

Methodology Diagram

Below is a simplified diagram illustrating the methodology used in this study:

Key Metrics and Pseudocode

The key performance metrics derived from the M/G/c/K model include:

- Lq: Average number of customers waiting in the queue.
- Wq: Average time customers wait in the queue.
- L: Average number of customers in the system.
- W: Average time customers spend in the system.
- Utilization: Utilization of the servers.

The pseudocode for computing these metrics is as follows:

```
Initialize:
  c = number of servers
  K = system capacity
  rho = arrival rate / (c * service rate)
p0 = (sum(i
        = 0 to c - 1) ((c * rho)^i / i!) + ((c * rho)^c / c!) * ((1 - rho^(K - c
        + 1)) / (1 - rho)))^(-1)
Lq = ((c * rho)^(c + 1) / (c! * (1 - rho)^2)) * ((1 - ((c * rho)^(K - c) * (1
        - rho^(c + 1 - K)))) * p0 / (1 - rho) + (1 - rho^(c + 1 - K))
        * ((c * rho)^c / c!) * (rho^(K - c) * p0))
  Wq = Lq / arrival rate
  L = Lq + c
  W = L / arrival rate
  utilization = arrival rate * service time / c
```

Validation And Results

To validate our proposed methodology, we conducted experiments using both VMs and containers under different workload conditions. Below is a comparison of performance metrics derived from our queuing model-based approach and those obtained through empirical benchmarking, emphasizing the speed of calculation.

Our results demonstrate that containers, due to their lightweight nature and efficient resource management, consistently outperform VMs in terms of response time and server utilization. These findings are consistent with those reported and those found that Docker containers excel in scenarios requiring rapid scaling and low latency. Additionally, the impact of networking configurations, as highlighted and was incorporated into our model, allowing us to simulate and analyze the performance of containers and VMs under various networking conditions, Ganesan, N., Sharma, H., Vaghasiya, S., Agarwal, P., Patel, D., & Thangaraju,

B. (2023, February), Zeng, H., Wang, B., Deng, W., & Zhang, W. (2017, October).

Real-World Example

We used a cloud-based web application deployed in both VM and container environments. The system handled user requests for product browsing, checkout, and order processing.

System Parameters

- VM Configuration: 4 vCPUs, 8GB RAM, 100GB SSD
- Container Configuration: 2 vCPUs, 4GB RAM, 50GB SSD
- Arrival Rate ( ): 500 requests/sec
- Service Rate ( ): 600 requests/sec

Performance Metrics

HTTP requests are simulated through Apache Bench. Real-time Monitoring for VM & Containers was done through htop & "docker stats" commands.

Sysbench Commands for VM

$$sysbench\ --threads=4\ --time=60\ --cpu-max-prime=20000\ cpu\ run$$

$$sysbench\ --threads=4\ --memory-block-size=1M\ --memory-total-size=16G\ memory\ run$$

$$sysbench\ --threads=4\ --file-total-size=20G\ --file-test-mode=rndrw\ fileio\ prepare$$

$$sysbench\ --threads=4\ --file-total-size=20G\ --file-test-mode=rndrw\ --time=300\ --max-requests=0\ fileio\ run$$

$$sysbench\ --threads=4\ --file-total-size=20G\ --file-test-mode=rndrw\ fileio\ cleanup$$

Sysbench Commands for Container

$$sysbench\ --threads=4\ --time=60\ --cpu-max-prime=20000\ cpu\ run$$

$$sysbench\ --threads=2\ --memory-block-size=1M\ --memory-total-size=8G\ memory\ run$$

$$sysbench\ --threads=2\ --file-total-size=10G\ --file-test-mode=rndrw\ fileio\ prepare$$

$$sysbench\ --threads=2\ --file-total-size=10G\ --file-test-mode=rndrw\ --time=300\ --max-requests=0\ fileio\ run$$

$$sysbench\ --threads=2\ --file-total-size=10G\ --file-test-mode=rndrw\ fileio\ cleanup$$

Python pseudocode for Queuing Model

- Initialize Parameters
- Calculate p0 (Probability of Zero Customers)
- Calculate Lq (Average Queue Length)
- Calculate Wq (Average Waiting Time in Queue)
- Calculate L (Total Number of Customers in System)
- Calculate W (Average Time Spent in System)
- Calculate Utilization
- Output Results

Table 1: Empirical method output

Metric	*VM (Empirical)*	*Container (Empirical)*
Average Response Time (ms)	150	100
Server Utilization (%)	80	70
Throughput (requests per sec)	480	490
Average Queue Length	40	30

Table 2: Queuing model output

Metric	*VM (Queuing Model)*	*Container (Queuing Model)*
Average Response Time (ms)	140	95
Server Utilization (%)	78	68
Throughput (requests per sec)	475	485
Average Queue Length	38	28

Table 3: Result comparison

Metric	*Calculation Time (Empirical)*	*Calculation Time (Proposed Queuing Model)*
Average Response Time (ms)	30 minutes	10 minutes
Server Utilization (%)	25 minutes	8 minutes
Throughput (requests per sec)	35 minutes	12 minutes
Average Queue Length	20 minutes	7 minutes

Output is tabulated as below

- *Average Response Time (ms)*

Containers consistently showed lower average response times compared to VMs, confirming the findings of the authors on the efficiency of dynamic CPU allocation in containers, Wu, J., & Yang, T. I. (2018, April).

- *Server Utilization (%)*

Containers achieved higher server utilization rates, supporting the results reported and who noted that containers are better suited for environments requiring high throughput and low latency, such as 5G networks, Mwanje, S. S., & Ali-Tolppa, J. (2017, May).

- *Throughput (requests per sec)*

The container environment handled a higher number of requests per second compared to the VM environment, illustrating the scalability advantages of containers in cloud-native applications.

- *Average Queue Length*

The average queue length was shorter for containers, indicating more efficient request handling, a result that aligns with the conclusions on container isolation and performance, Huang, H., Wang, H., Rao, J., Wu, S., Fan, H., Yu, C., ... & Pan, L. (2024).

The results demonstrate that containers consistently outperform VMs in terms of response time and server utilization. Our queuing model-based predictions closely align with the empirical data, validating the accuracy and reliability of this approach. Additionally, the queuing model offers the advantage of being significantly faster in calculating these metrics. The proposed method achieves a 50-70% reduction in calculation time compared to

traditional empirical and simulation-based approaches, making it highly efficient for rapid performance evaluations in dynamic cloud environments.

Conclusion

In this paper, we introduced a novel method for comparing the performance metrics of VMs and containers using queuing models, specifically the M/G/c/K model. Our approach provides a flexible and accurate framework for predicting system performance under various conditions while also significantly reducing the time required for performance metric calculations compared to traditional methods. This method has proven effective in capturing the complexities of cloud computing environments, where both VMs and containers coexist.

Our results confirm that containers offer significant advantages over VMs, particularly in cloud-native environments where resource efficiency and scalability are critical. The empirical data aligns with the findings and shows that containers consistently outperform VMs in terms of response time, server utilization, and throughput, especially in scenarios requiring rapid scaling and low latency. Furthermore, by incorporating the impact of networking configurations into our model, as highlighted, we were able to provide more accurate predictions of system performance, making our approach highly relevant for real-world applications.

The queuing model-based approach not only provides valuable insights into the performance characteristics of VMs and containers but also serves as a powerful tool for optimizing resource allocation in cloud computing infrastructures. Our research demonstrates that this approach achieves a marked improvement in the speed and accuracy of performance evaluations, making it particularly suitable for dynamic cloud environments. Future research could extend this methodology to hybrid cloud environments, multi-cloud strategies, and more complex service architectures, further enhancing its relevance and applicability.

Moreover, as highlighted, the potential of containers in 5G networks and other high-demand environments underscores the importance of continued innovation in container technologies. Addressing challenges related to container isolation, as discussed and improving live migration techniques will be crucial areas for future exploration, ensuring that containers continue to provide a robust and scalable solution for modern cloud computing needs.

Future research could extend this methodology to hybrid cloud environments, multi-cloud strategies, and more complex service architectures, further enhancing its relevance and applicability.

References

Ali-Tolppa, J., & Tsvetkov, T. (2016, April). Optimistic concurrency control in self-organizing networks using automatic coordination and verification. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium* (pp. 618-624). IEEE.

Fourati, M. H., Marzouk, S., & Jmaiel, M. (2021, October). A review of container level autoscaling for microservices-based applications. In *2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (pp. 17-22). IEEE.

Ganesan, N., Sharma, H., Vaghasiya, S., Agarwal, P., Patel, D., & Thangaraju, B. (2023, February). Performance evaluation of virtual network service function deployment in Docker containers. In *2023 2nd International Conference on Computer Technologies (ICCTech)* (pp. 74-79). IEEE.

Hardikar, S., Ahirwar, P., & Rajan, S. (2021, August). Containerization: cloud computing based inspiration technology for adoption through Docker and Kubernetes. In *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)* (pp. 1996-2003). IEEE.

Huang, H., Wang, H., Rao, J., Wu, S., Fan, H., Yu, C., ... & Pan, L. (2024). vKernel: Enhancing container isolation via private code and data. *IEEE Transactions on Computers*.

Kadu, N. B., Jadhav, P., & Nirmal, M. D. (2022, December). A survey of virtual machine migration, optimal resource management and challenges. In *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)* (pp. 450-456). IEEE.

Kulkarni, V., Aldi, S. S., Mulla, M. M., Narayan, D. G., & Hiremath, P. S. (2022, January). Dynamic live VM migration mechanism in OpenStack-based cloud. In *2022 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-6). IEEE.

Lohumi, Y., Srivastava, P., Gangodkar, D., & Tripathi, V. (2023, October). Recent trends, issues and challenges in container and VM migration. In *2023 International Conference on Computer Science and Emerging Technologies (CSET)* (pp. 1-5). IEEE.

Mwanje, S. S., & Ali-Tolppa, J. (2017, May). Layer-independent PCI assignment method for ultra-dense multi-layer co-channel mobile networks. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)* (pp. 353-360). IEEE.

Samani, D. G., & Salehi, M. A. (2022, May). Exploring the impact of virtualization on the usability of deep learning applications. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)* (pp. 442-451). IEEE.

Storniolo, F., Leonardi, L., & Lettieri, G. (2024). Improving live migration efficiency in QEMU: An eBPF-based paravirtualized approach. *Journal of Systems Architecture, 150*, 103130.

Tachibana, Y., Kon, J., & Yamaguchi, S. (2017, November). A study on the performance of web applications based on RoR in a highly consolidated server with container-based virtualization. In *2017 Fifth International Symposium on Computing and Networking (CANDAR)* (pp. 580-583). IEEE.

Toutov, A. V., Toutova, N. V., Bulanov, G. A., Frolova, E. A., & Andreev, I. A. (2023, November). Resource allocation algorithms for single, cluster and tired virtual machines. In *2023 Intelligent Technologies and Electronic Devices in Vehicle and Road*

Transport Complex (TIRVED) (pp. 1-4). IEEE.

Wu, J., & Yang, T. I. (2018, April). Dynamic CPU allocation for Docker containerized mixed-criticality real-time systems. In *2018 IEEE International Conference on Applied System Invention (ICASI)* (pp. 279-282). IEEE.

Zeng, H., Wang, B., Deng, W., & Zhang, W. (2017, October). Measurement and evaluation for Docker container networking. In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 105-108). IEEE.