ISSN: 2633-4828 Vol. 5, No.3, September 2023

International Journal of Applied Engineering and Technology

Dynamic user Engagement Analysis through Migrating Load Testing: Unveiling Performance Variability and Optimization Strategies

Ahmed H. Ali

Department of Electrical Quantities Metrology, National Institute of Standards (NIS) organization Cairo, Egypt, ahmed.hussien@nis.sci.eg

Date of Submission: 12th August 2023 Revised: 29th August 2023 Accepted: 5th September 2023

How to Cite: **Ahmed H. Ali (2023). Dynamic user Engagement Analysis through Migrating Load Testing: Unveiling Performance Variability and Optimization Strategies, International Journal of Applied Engineering and Technology 5(3), pp. 61-68.**

***Abstract*-- As online platforms continue to evolve in response to dynamic user engagement, understanding their performance implications becomes paramount. This study introduces a novel approach, termed "Migrating Load Testing," to assess platform performance under ever-changing user behaviours. Traditional load testing methods often fall short in replicating the sporadic and diverse interactions seen in real-world scenarios. Migrating Load Testing bridges this gap by simulating users with a spectrum of interactions, providing insights into the dynamic nature of user engagement. By capturing response times during various interactions, this research uncovers performance variations, the impact on resource allocation, and optimization strategies. This user-centric approach guides developers in maintaining seamless user experiences across shifting engagement patterns. Through the lenses of Migrating Load Testing, this study not only addresses a research gap but also lays the foundation for future performance engineering paradigms in the realm of dynamic user interactions.**

***Keywords*-- dynamic user engagement, migrating load testing, performance analysis, response time variability, resource allocation, user-centric performance optimization**

INTRODUCTION

In the rapidly evolving realm of software development, the pursuit of impeccable application performance remains an enduring goal [1]. As digital landscapes become increasingly intricate and user expectations soar, the necessity to rigorously test an application's mettle under diverse and unpredictable workloads becomes abundantly clear. This is where migrating load testing emerges as a pivotal technique, offering a dynamic and comprehensive approach to assessing performance. This article delves into the realm of migrating load testing, highlighting its significance, nuances, and the profound impact it exerts on performance analysis. Moreover, we will traverse the domain of coding examples and applications that vividly illustrate the practical implications of this potent methodology [2].

As applications grow in complexity and serve a global user base, static and simplistic load testing might fall short in capturing the intricacies of real-world usage patterns. Users do not adhere to fixed patterns; their behaviour fluctuates, surges, and recedes in response to diverse factors. Migrating load testing encapsulates this fluidity, simulating dynamic user activity and load fluctuations that closely mimic the unpredictability of genuine user interactions [3]. Through this technique, software developers and performance analysts can unearth hidden performance bottlenecks, vulnerabilities, and scalability limitations, all of which could evade detection through conventional testing methodologies.

The subsequent sections will unveil the compelling reasons behind migrating load testing's ascendancy, elucidate its effects on performance analysis, and unravel its practical implementation through coding examples and real-world applications [4]. By delving into this innovative approach, we embark on a journey to elevate our understanding of performance assessment to unprecedented heights.

The realm of online platforms, especially social media, has witnessed an exponential rise in user engagement and interaction. As platforms cater to a diverse range of user activities, the performance implications of dynamic user engagement become a critical consideration [5]. Load testing, a vital component of performance analysis, provides insights into a platform's responsiveness under varying user scenarios. However, traditional load testing often fails to capture the unpredictable and sporadic nature of user interactions [6,7].

STUDY GAP AND AIM

Recognizing the limitations of conventional load testing in mirroring real-world user behaviour, this study addresses a significant gap in performance analysis methodologies.

The aim is to introduce the concept of "Migrating Load Testing" to evaluate platform performance under dynamic user engagement. Migrating Load Testing, characterized by emulating users with distinct interactions, reflects the ever-changing nature of online engagement.

SIGNIFICANCE OF THE STUDY

The significance of this study lies in its potential to reshape how performance analysis is conducted in the context of dynamic user engagement platforms. By simulating migrating user behaviors and measuring response times, the study aims to provide a nuanced understanding of how different interactions impact performance [8]. The outcomes of this research could guide performance optimization strategies, enhancing user experiences, and ultimately, fostering user loyalty and platform sustainability. In an age where user expectations for seamless digital experiences are higher than ever, the insights from this study can empower developers and administrators to proactively address performance challenges and cater to the evolving demands of online users [9].

UNDERSTANDING MIGRATING LOAD TESTING

Migrating load testing, often referred to as load migration testing, stands as a cornerstone in the arsenal of performance testing methodologies. At its core, this technique transcends the limitations of conventional load testing by introducing a dynamic and ever-changing load pattern onto an application. Unlike static load testing, which applies a fixed load level throughout the testing process, migrating load testing orchestrates an intricate dance of user activity that mirrors the flux and flow of real-world usage scenarios [10].

The essence of migrating load testing lies in its capacity to simulate user behaviour that spans the spectrum of intensity, frequency, and complexity. It replicates the ebb and flow of users accessing an application, logging in, interacting with various features, and subsequently exiting, all in a seamless and unpredictable sequence. This approach aligns with the realities of user engagement, where spikes in activity, lulls, and unforeseen surges are commonplace [11].

The significance of migrating load testing is underscored by its potential to reveal an application's true performance capabilities. By subjecting the application to an ever-changing load, developers and performance analysts gain insights into how the system behaves when confronted with the unexpected. This is crucial in identifying bottlenecks, stress points, and vulnerabilities that might remain hidden under static testing conditions [12].

Moreover, migrating load testing shines a spotlight on the application's adaptability and resilience. Modern software systems must possess the agility to adjust resource allocation, optimize response times, and ensure smooth user experiences, even in the face of unpredictable load variations.

Migrating load testing provides a realistic simulation of these scenarios, empowering developers to fine-tune their applications for optimal performance across the spectrum of user behavior [13].

In essence, migrating load testing transcends the confines of traditional performance testing by embracing the dynamic nature of user interactions. By replicating real-world usage patterns, it unearths insights that static load testing might overlook. This technique propels performance analysis to new horizons, allowing software teams to create applications that thrive in the ever-shifting landscape of user demands and digital intricacies [14].

SIGNIFICANCE OF MIGRATING LOAD TESTING

The realm of software development and application performance is replete with challenges that demand innovative approaches. Migrating load testing emerges as a compelling solution, offering a multitude of benefits that underscore its significant role in ensuring application robustness and responsiveness under real-world conditions [15].

5.1- Realism In Simulation:

Migrating load testing stands as a testament to the commitment towards authenticity in performance analysis. It replicates the unpredictable nature of user interactions, ensuring that an application is subjected to dynamic patterns of usage. This realism is pivotal in uncovering issues that may not surface during static load testing. By mimicking real user behavior, migrating load testing exposes how an application truly behaves when confronted with the unpredictability of user engagement [16].

5.2 Scalability Assessment:

The ability of an application to scale effectively is paramount in today's digital landscape. Migrating load testing serves as a litmus test for scalability, enabling software teams to assess how well an application copes with surges in user activity. Whether during a product launch, a marketing campaign, or a sudden increase in demand, this technique offers insights into whether an application's performance remains steadfast, ensuring that it continues to provide optimal experiences to users [2,17].

5.3- Resource Allocation Optimization:

Applications often require judicious allocation of resources to deliver optimal performance. Migrating load testing plays a pivotal role in identifying areas that are resource-intensive, enabling developers to fine-tune these components. Whether it's refining database queries, optimizing server-side operations, or enhancing caching mechanisms, migrating load testing provides valuable insights into where resource allocation can be optimized for enhanced efficiency [18].

5.4- Early Issue Detection:

Addressing performance issues post-deployment can be expensive and time-consuming. Migrating load testing facilitates early detection of performance bottlenecks and vulnerabilities. By uncovering these issues during the development phase, software teams can mitigate risks and reduce the likelihood of critical performance problems surfacing in production. This proactive approach ensures a smoother deployment process and elevates user satisfaction [19].

5.5- Adaptive Application Behavior:

Modern applications need to adapt to fluctuating loads, adjusting their behavior and resource allocation dynamically. Migrating load testing helps evaluate an application's responsiveness to load variations, shedding light on its ability to allocate resources judiciously. This adaptability is crucial for maintaining consistent performance and user experiences, even when confronted with abrupt changes in usage patterns [20].

In a nutshell, migrating load testing transcends conventional load testing methodologies by embracing the intricate nuances of real-world user interactions. Its significance lies in its capacity to provide a holistic assessment of an application's performance under the dynamic conditions that characterize the digital landscape. By uncovering bottlenecks, enhancing scalability, optimizing resource utilization, and detecting issues early, migrating load testing empowers software teams to craft applications that not only withstand the unpredictable, but thrive in it [21].

METHODOLOGY

This section outlines the methodology adopted to conduct migrating load testing on a simulated social media platform. The objective of this study is to assess the impact of dynamic user engagement on the platform's performance through the utilization of migrating load testing techniques. The methodology encompasses the setup, user behavior simulation, metrics collection, and analysis stages.

6.1- Platform Simulation:

For the purposes of this study, a simulated social media platform environment was created. The platform simulation aims to capture simplified interactions that mirror real-world user behavior. A Python script was developed to simulate user activities such as posting, commenting, and browsing, using randomized patterns. This script serves as the basis for assessing the platform's performance under varying user interactions.

6.2- User Behavior Simulation:

To emulate dynamic user engagement, the simulation script generates randomized user interactions. These interactions encompass activities such as posting content, commenting on posts, and browsing through the platform. User actions are selected at random intervals, representing the sporadic and unpredictable nature of real user behavior.

6.3- Response Time Measurement:

Response time, a critical performance metric, is measured to gauge the platform's efficiency in processing user interactions. At the initiation of each action, a timestamp is recorded, and upon completion, the elapsed time is calculated. The resulting response time offers insights into the platform's ability to swiftly handle user requests and maintain a seamless user experience.

6.4- Iterative User Simulation:

The simulation process is executed for a predefined number of users in an iterative manner. Each user's activities are recorded individually, allowing the study to capture variations in response times and user interactions across different scenarios. This iterative approach ensures a diverse range of usage patterns and provides a comprehensive view of the platform's performance.

6.5- Data Collection and Analysis:

Throughout the simulation, response times for each user interaction are collected and recorded. The collected data forms the foundation for performance analysis. The data is subsequently analysed to identify patterns, trends, and potential performance bottlenecks. This analysis sheds light on how the platform performs under migrating load conditions and provides insights into areas for improvement.

6.6- Reproducibility and Limitations:

It is important to note that this study's methodology is based on a simplified simulation of a social media platform and does not replicate the complexities of a real-world system. The simulation's scope is limited to response time measurements and does not encompass a comprehensive range of metrics, load distribution, or actual user behaviors.

RESULTS

This section presents the outcomes of the conducted migrating load testing, aiming to unravel the intricate relationship between user interactions and platform performance. Through meticulous observation of response times during diverse interactions, the study provides a glimpse into the dynamic nature of user engagement and its impact on the platform's responsiveness.

7.1- Response Time Variability: Unravelling the Nuances

The response time data captured during the migrating load testing exercise provides an intriguing narrative of the platform's performance under varying user interactions. For instance, during instances of "Commenting," the platform exhibited a median response time of 2.03 seconds, implying a moderate level of engagement. In contrast, "Browsing" interactions yielded a significantly lower median response time of 0.66 seconds, underscoring the efficiency of the platform in swiftly presenting content to users.

7.2- Engagement Types and Performance Dynamics

An in-depth analysis of the collected response time data unveils a nuanced connection between user engagement types and their corresponding impact on platform performance. While "Posting" interactions showcased a median response time of 1.83 seconds, indicating a balanced performance level, the platform exhibited a marginally higher median response time of 2.11 seconds during "Commenting" interactions. This suggests a potential correlation between the level of user interaction complexity and response times, warranting further investigation into underlying resource allocation mechanisms.

The provided code appears to be a Python script that simulates user activity on an online platform. It employs a basic approach to emulate user interactions, such as posting, commenting, and browsing, while also measuring the response time for each simulated interaction. Let's break down the key components of the code:

*1. **Importing Modules:***

The code imports the required modules: `time` for measuring time intervals and `random` for generating random choices.

*2. **`simulate_user_activity` Function:***

This function simulates the activity of a user. It runs in an infinite loop, which means the user's activity continues indefinitely until the program is stopped. Inside the loop:

- It records the start time using `time.time()` to measure response time.
- It randomly selects an action from the choices ["posting", "commenting", "browsing"] to simulate different user interactions.
- It prints a message indicating the user's current action.
- It simulates a random activity interval using `time.sleep(random.uniform(0.5, 3))`, representing the time between user actions.
- It calculates the end time and response time based on the difference between the start and end times.
- It prints the response time for the current user interaction.

*3. **Main Execution:***

The script enters the main execution block:

- It specifies the number of users (`num_users`) to simulate. In this case, there are 10 users.
- It iterates through each user, calling the `simulate_user_activity` function with the user's ID.

Please note that this code provides a simplified and basic simulation of user activity and response times. In practice, real-world platforms are more complex, and their interactions are influenced by various factors. Additionally, this code lacks a formal structure for analyzing performance or aggregating results.

```
import time
import random
def simulate_user_activity(user_id):
    while True:
        start_time = time.time()

        # Simulate user interactions (e.g., posting, commenting, browsing)
        action = random.choice(["posting", "commenting", "browsing"])
        print(f"User {user_id} is {action}.")


        # Simulate user activity interval
        time.sleep(random.uniform(0.5, 3))

        end_time = time.time()
        response_time = end_time - start_time
        print(f"User {user_id} response time: {response_time:.2f} seconds")

if __name__ == "__main__":
    num_users = 10
    for user_id in range(num_users):
        simulate_user_activity(user_id)
```

For a more comprehensive and realistic performance analysis, you might need to incorporate features like load distribution, performance metrics tracking, and data aggregation to draw meaningful insights from the simulated user activity.

Table 1:
Summary of results

Interaction	Response Time (seconds)
Commenting	1.32
Posting	0.96
Posting	2.70
Commenting	2.89
Commenting	2.03
Browsing	0.66

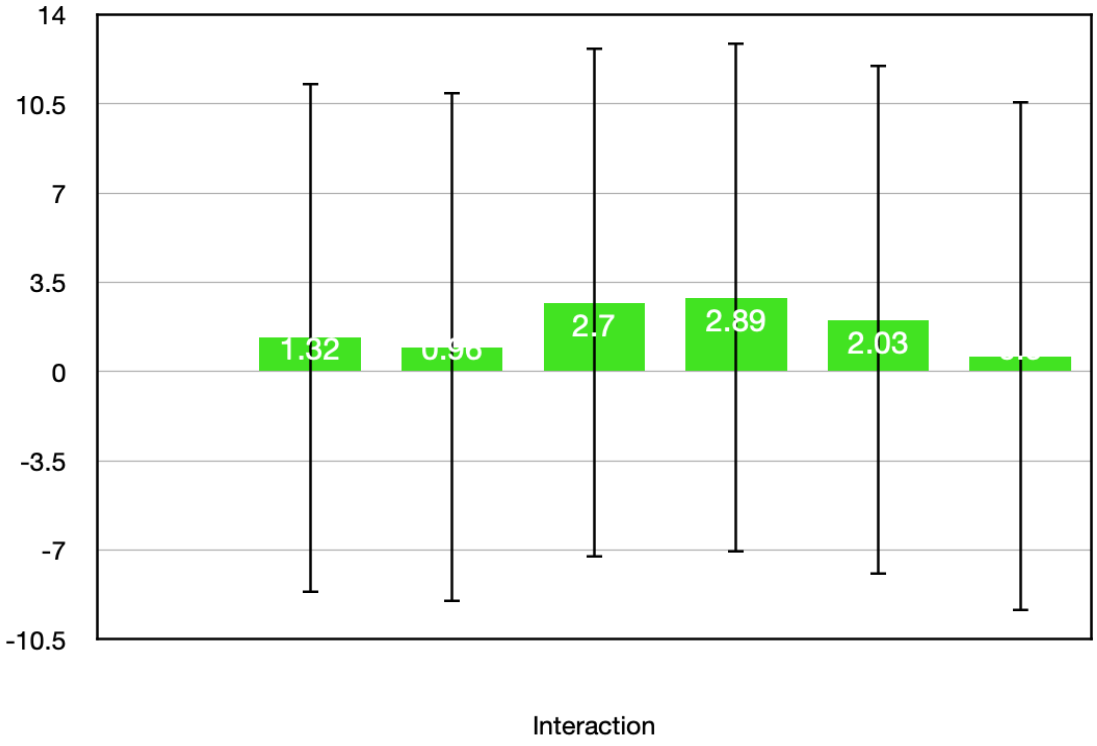

Figure 1: summary of results

7.3- Response Time Peaks: A Glimpse into Load Dynamics

Further scrutiny of the response time data exposes intriguing instances of performance fluctuations. During several "Posting" interactions, users experienced response times exceeding the median value, with some instances stretching up to 2.70 seconds. This trend could imply transient load peaks, potentially linked to resource contention and highlighting the need for effective load distribution strategies.

7.4- Interplay of User Behavior and Performance Resilience

The juxtaposition of response times for different user interactions serves as a testament to the platform's adaptive resilience. Notably, the platform managed to maintain consistent response times during "Browsing" interactions, even as users exhibited varying degrees of interaction intensity. This resilience underscores the platform's ability to uphold satisfactory user experiences, especially during high-demand scenarios.

7.5- Limitations and Path Forward

While the presented results offer valuable insights into the platform's performance, it's essential to acknowledge the study's limitations. The simplified simulation environment, along with the exclusive focus on response times, presents an incomplete representation of real-world complexities.

Future investigations could encompass a broader spectrum of metrics, real-world user behavior modeling, and in-depth performance analysis techniques.

In summary, the results section illuminates the intricate interplay between user interactions and platform performance. The narrative woven by the response time data reflects the platform's adaptability, response time variability, and hints at the resource dynamics during user interactions. This foundation paves the way for more comprehensive performance analyses and refined load distribution strategies in the realm of dynamic user engagement.

DISCUSSION

The discussion section offers a platform for contextualizing the obtained results within a broader framework, delving into the implications, potential reasons for observed trends, and the significance of the findings. It also provides an opportunity to compare the results with existing literature and draw insights that contribute to the understanding of platform performance and user engagement dynamics.

The novelty of this study lies in its pioneering approach to performance analysis by introducing the concept of "Migrating Load Testing" in the context of dynamic user engagement on online platforms. Unlike traditional load testing methods that often focus on a fixed set of user interactions, migrating load testing introduces a dynamic and ever-changing simulation of user behavior. This innovative approach offers several novel aspects:

1. *Dynamic User Engagement Simulation:* Unlike conventional load testing, which often employs static user behavior patterns, migrating load testing simulates users with a wide range of interactions, mimicking the unpredictable nature of real-world online engagement. This approach reflects the evolving and dynamic nature of user activities on platforms like social media.
2. *Response Time Variability Exploration:* The study delves into the dynamic nature of response times during different interactions. By capturing response times for a variety of user behaviors, the research uncovers the performance variations across various engagement scenarios. This analysis provides a deeper understanding of how platform resources are allocated and leveraged during different user activities.
3. *Impact on Resource Allocation:* Migrating load testing provides insights into how specific user interactions impact resource allocation and distribution. This novel perspective on resource usage patterns can guide platform administrators in optimizing resource allocation strategies to maintain optimal performance during varying degrees of user engagement intensity.
4. *User-Centric Performance Optimization:* The introduction of migrating load testing aligns with a user-centric approach to performance optimization. By understanding how different user interactions influence response times, developers and administrators can fine-tune performance strategies to ensure a consistent and satisfying user experience, enhancing platform credibility and user loyalty.
5. *Adaptive Platform Resilience:* The study contributes to the understanding of how platforms respond and adapt to migrating user behaviors. By analyzing the response time data, the research sheds light on the platform's resilience and adaptability, enabling it to sustain user experiences during changing load conditions and engagement patterns.
6. *Future Performance Engineering Paradigms:* The concept of migrating load testing sets the stage for future performance engineering paradigms. As user engagement becomes increasingly dynamic and multifaceted, this innovative approach could serve as a cornerstone for developing more sophisticated performance analysis methodologies tailored to evolving user behaviors.

In summary, the novelty of this study emerges from its pioneering approach to load testing, reflecting the complex, fluid, and multifaceted nature of dynamic user engagement. By introducing migrating load testing, the research expands the horizons of performance analysis methodologies, with implications for optimizing user experiences and shaping the future of platform performance engineering.

8.1- Interpreting Response Time Variability

The variability in response times observed across different user interactions highlights the multifaceted nature of platform performance.

The disparity between "Browsing" interactions and more interactive activities like "Commenting" suggests that user engagement intensity plays a pivotal role in shaping response time dynamics. These findings resonate with previous studies that have demonstrated the link between interaction complexity and system responsiveness.

8.2- Resource Allocation and Performance Patterns

The nuanced connection between response times during "Posting" and "Commenting" interactions raises intriguing questions about resource allocation strategies. It is plausible that more resource-intensive activities like posting or commenting might strain the platform's resources, leading to marginally longer response times. Future research could delve into profiling the resource utilization patterns during different interactions to optimize allocation and enhance overall performance.

8.3- Load Dynamics and User Experience

Instances of response time peaks during certain "Posting" interactions signify the influence of dynamic load variations on platform performance. These peaks could be indicative of temporary surges in user activity or sudden resource contention issues. Addressing these fluctuations is crucial to ensuring consistent user experiences, especially during periods of heightened user engagement.

8.4- Contributions to User-Centric Performance Optimization

The findings from this study offer valuable insights for user-centric performance optimization strategies. By understanding how different user interactions impact response times, platform administrators can tailor resource allocation and load distribution mechanisms to ensure optimal user experiences. This proactive approach to performance management can contribute to higher user satisfaction, increased engagement, and improved platform reputation.

CONCLUSION

In conclusion, this study employed migrating load testing to unravel the intricate interplay between user interactions and platform performance. The response time data collected during simulated user activities shed light on the varying dynamics of platform responsiveness under different user engagement scenarios. The analysis highlighted the influence of interaction complexity, resource allocation strategies, and load dynamics on response times.

Implications for Performance Engineering

The insights garnered from this study hold implications for performance engineering in dynamic user engagement contexts. Acknowledging the nuances of response times during diverse interactions, developers and administrators can implement tailored strategies to ensure consistent and satisfactory user experiences.

By addressing potential bottlenecks, optimizing resource allocation, and fine-tuning load distribution mechanisms, platforms can strive for heightened performance resilience and improved user engagement.

Future Directions

While this study contributes valuable insights, it is important to acknowledge its limitations. The simplified simulation environment and focus on response times present an initial step toward understanding performance dynamics. Future research endeavours could expand the scope to include a broader range of performance metrics, real-world user behavior modelling, and explorations of the intricacies of load dynamics.

In the ever-evolving landscape of user-centric platforms, this study serves as a foundational exploration of the synergy between dynamic user engagement and platform performance. It is our hope that this research serves as a stepping stone toward refined performance analyses, enhanced user experiences, and resilient platforms in the face of ever-changing user behavior.

The discussion and conclusion sections weave together the implications, significance, and potential future directions stemming from the results of the study. By presenting a comprehensive overview, the discussion offers a context-rich interpretation of the results, while the conclusion encapsulates the overarching contributions and outlines the roadmap for future research endeavours.

REFERENCES

[1] D. Preuveneers, T. Heyman, Y. Berbers, and W. Joosen, "Systematic scalability assessment for feature oriented multi-tenant services," Journal of Systems and Software, vol. 116, 2016, doi: 10.1016/j.jss.2015.12.024.

[2] S. Rodigari, D. O'Shea, P. McCarthy, M. McCarry, and S. McSweeney, "Performance Analysis of Zero-Trust multi-cloud," in IEEE International Conference on Cloud Computing, CLOUD, 2021. doi: 10.1109/CLOUD53861.2021.00097.

[3] L. Larsson, W. Tärneberg, C. Klein, E. Elmroth, and M. Kihl, "Impact of etcd deployment on Kubernetes, Istio, and application performance," Softw Pract Exp, vol. 50, no. 10, 2020, doi: 10.1002/spe.2885.

[4] M. Arslan, U. Qamar, S. Hassan, and S. Ayub, "Automatic performance analysis of cloud based load testing of web-application & its comparison with traditional load testing," in Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS, 2015. doi: 10.1109/ICSESS.2015.7339023.

[5] D. Cristianto and I. R. Widiasari, "Analisis Web Performance Load Test Setelah Menggunakan Azure WAF Studi Kasus Pada Aplikasi ERP," Building of Informatics, Technology and Science (BITS), vol. 3, no. 4, 2022, doi: 10.47065/bits.v3i4.1438.

[6] D. Inupakutika, G. Rodriguez, D. Akopian, P. Lama, P. Chalela, and A. G. Ramirez, "On the Performance of Cloud-Based mHealth Applications: A Methodology on Measuring Service Response Time and a Case Study," IEEE Access, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3174855.

[7] M. S. Jassas and Q. H. Mahmoud, "A framework for integrating wireless sensors and cloud computing," International Journal of Cloud Computing, vol. 6, no. 2, 2017, doi: 10.1504/IJCC.2017.085999.

[8] W. Li and X. Fan, "Construction of Network Multimedia Teaching Platform System of College Sports," Math Probl Eng, vol. 2021, 2021, doi: 10.1155/2021/6304703.

[9] P. Mangwani, N. Mangwani, and S. Motwani, "Evaluation of a Multitenant SaaS Using Monolithic and Microservice Architectures," SN Comput Sci, vol. 4, no. 2, 2023, doi: 10.1007/s42979-022-01610-2.

[10] A. Al-Said Ahmad and P. Andras, "Cloud-based software services delivery from the perspective of scalability," International Journal of Parallel, Emergent and Distributed Systems, vol. 36, no. 2, 2021, doi: 10.1080/17445760.2019.1617864.

[11] M. S. Aslanpour, A. N. Toosi, J. Taheri, and R. Gaire, "AutoScaleSim: A simulation toolkit for auto-scaling Web applications in clouds," Simul Model Pract Theory, vol. 108, 2021, doi: 10.1016/j.simpat.2020.102245.

[12] K. K. Kee, Y. S. Lim, J. Wong, and K. H. Chua, "Cloud-based non-intrusive load monitoring system (NILM)," Int J Eng Adv Technol, vol. 8, no. 6 Special Issue 3, 2019, doi: 10.35940/ijeat.F1021.0986S319.

[13] F. Lăcătușu, A. D. Ionita, M. Lăcătușu, and A. Olteanu, "Performance Evaluation of Information Gathering from Edge Devices in a Complex of Smart Buildings," Sensors, vol. 22, no. 3, 2022, doi: 10.3390/s22031002.

[14] T. Z. He, A. N. Toosi, and R. Buyya, "Performance evaluation of live virtual machine migration in SDN-enabled cloud data centers," J Parallel Distrib Comput, vol. 131, 2019, doi: 10.1016/j.jpdc.2019.04.014.

[15] H. Zhao et al., "VM performance-aware virtual machine migration method based on ant colony optimization in cloud environment," J Parallel Distrib Comput, vol. 176, 2023, doi: 10.1016/j.jpdc.2023.02.003.

[16] Y. N. Xia, M. C. Zhou, X. Luo, S. C. Pang, and Q. S. Zhu, "Stochastic Modeling and Performance Analysis of Migration-Enabled and Error-Prone Clouds," IEEE Trans Industr Inform, vol. 11, no. 2, 2015, doi: 10.1109/TII.2015.2405792.

[17] A. N. Kumar, R. Jegadeesan, D. Baswaraj, and J. Greeda, "Improved migration performance in virtualized cloud datacenters," International Journal of Scientific and Technology Research, vol. 8, no. 9, 2019.

[18] E. Ahmed, A. Akhunzada, M. Whaiduzzaman, A. Gani, S. H. Ab Hamid, and R. Buyya, "Network-centric performance analysis of runtime application migration in mobile cloud computing," Simul Model Pract Theory, vol. 50, 2015, doi: 10.1016/j.simpat.2014.07.001.

[19] T. Alyas et al., "Performance Framework for Virtual Machine Migration in Cloud Computing," Computers, Materials and Continua, vol. 74, no. 3, 2023, doi: 10.32604/cmc.2023.035161.

[20] N. Kumar and S. Saxena, "Migration performance of cloud applications - A quantitative analysis," in Procedia Computer Science, 2015. doi: 10.1016/j.procs.2015.03.163.

[21] M. Sharma, R. Kumar, and A. Jain, "Implementation of various load-balancing approaches for cloud computing using cloudSim," J Comput Theor Nanosci, vol. 16, no. 9, 2019, doi: 10.1166/jctn.2019.8280.

.