

Original Article

The Effectiveness of Load and Performance Testing on Application Scalability

Mohnish Neelapu

Received Date: 25 July 2024

Revised Date: 30 August 2024

Accepted Date: 23 September 2024

Abstract: Modern applications need performance testing to determine their scalability and efficiency because it verifies that rising user loads will not degrade performance capabilities. The research analyzes multiple techniques used for load and performance testing through stress testing and endurance testing and spike testing to study system conduct during various workload scenarios. The study assesses performance metrics through the utilization of Apache JMeter and LoadRunner and Gatling and k6 testing tools to evaluate response time as well as throughput and resource usage. Experiments validate system performance alterations that occur when multiple users access the system at once while exposing optimization strategies for these situations. The effectiveness of performance testing tools becomes clear through their analysis when assessing actual real-world implementation. The research outcomes underscore automated testing frameworks as key elements for maintaining application dependability alongside providing consistent user interactions.

Keywords: Load Testing, Performance Testing, Scalability, Response Time, Resource Utilization, Automation.

I. INTRODUCTION

The development of modern software systems requires high performance together with scalability because it directly affects user experience quality. Applications must handle growing user traffic while maintaining their performance efficiency which stands as their primary operational concern. Software evaluation under various demand levels and bottleneck identification as well as resource optimization achieves its objectives through Load and performance testing [6]. These testing approaches serve organizations to boost system dependability and minimize disruptions while fulfilling their Service Level Agreements (SLAs). The rising usage of cloud computing and microservices along with distributed architecture systems makes performance are testing more demanding so organizations need specialized tools and techniques to properly evaluate system functions. The reliability validation process enabled by performance testing helps to demonstrate application dependability and enables peak load testing to maintain service quality. Businesses that depend on web applications and mobile platforms together with cloud services need to maintain consistent performance across different environments because it has become essential [7]. Up-to-date performance testing approaches rely on automation alongside AI since they fail to duplicate actual customer scenarios. This makes modern applications demand these testing methods. In addition, organizations are challenged with incorporating performance testing into agile and DevOps processes to support continuous monitoring and optimization of application scalability.

Application scalability is the ability of the system to respond to the increasing load in a way to accommodate the change in load in a way that does not influence the performance of the system. Response Time, Throughput degradation can be avoided in such a way that such a scalable application can deal with increasing number of user requests, increase in data processing demand and concurrent transaction. Modern software applications (especially in cloud based services, e-Commerce platform and enterprise solutions) require scalability due to their fluctuating user traffic. Before the deployment of your application and with the aim of ensuring scalability limits, the perfect infrastructure optimization, and maintain a high quality user experience in such diverse environments, performance testing is of crucial importance. Directly, Scalability is not about being able to handle more users, it is about making a resource utilization optimization, building up in a fault tolerant way, and keeping the performance constant regardless of introduced workloads [8]. In cloud environments it is necessary that in the presence of traffic surges applications also dynamically allocate, and deallocate resources while incurring minimum costs or compromising on the service levels. For instance, in the realm of online platforms in high traffic such as e-Commerce websites, or even video streaming services, a poor scale to match demand can affect not only the user experience, but also result in loss, and reputational damage [9]. Performance testing plays an indispensable role in measuring and guaranteeing scalability, identifying bottlenecks, gauging infrastructure toughness, and enhancement of system parts. Automated performance testing frameworks enable businesses to increase efficiency and detect performance issues that will affect end users proactively. In addition, towards the present, as



microservices, containerized applications, and distributed computing become the norm, the modern scalability testing also needs to evolve to handle the newer complexities involved in system performance [10].

A. Research Objectives and contribution

The primary objectives of this research are:

- To study the most recent advancements regarding performance and scalability testing methodologies.
- To compare conventional and contemporary performance testing methods and how they affect application scalability.
- To identify key challenges that arise in the implementation of scalability focused performance testing strategy.
- A framework of proposing an optimized one combining automation, real time assessment and adaptive benchmarking techniques to performance test.

This study contributes to the area of research by delivering a critical evaluation of scalability testing approaches, proposing a new model for automatic performance assessment, and filling the gap in existing scalability evaluation methods through research.

II. LITERATURE REVIEW

Many studies have been present on developing the performance testing methodologies of modern applications to improve their scalability. There have been proposed, various approaches including automation testing, benchmarking, and frameworks. The major contributions of load and performance testing for application scalability are reviewed in this section. Avritzer et al. [1] present a methodology for automated scalability testing based on multivariate characterization and antipatterns detection of software performance. Specifically, they studied using system performance as they varied the load on the system. The proposal is proposed to be able to effectively detect the inefficiencies in the software architecture, with the goal for the system to become scalable. In particular, Gortázar et al. [2] conducted an investigation to identify load testing strategies for WebRTC applications for the lowest possible costs. The authors designed a new framework to simulate high volume concurrent users that guarantees the best resource utilization. It served as a valuable contribution towards the real time application scalability testing by highlighting the challenges of real time communication systems and associated solutions for area of performance degradation. Mungoli [3] discussed the role of distributed AI frameworks in improving the performance of deep learning by means of cloud computing is investigated. The idea behind the study was that cloud based infrastructures are able to train scalable AI model through distributing the workloads across multiple nodes. The result of which underlined the significance of utilizing the cloud computing in performance testing and good optimization of AI Driven Applications. Henning and Hasselbring [4] suggested a method for configuring scalable cloud native applications. Based on this framework, the developers could assess scalability for the application in different deployment configurations and identify the compromises between resource allocation and application performance. It helped to a large extent in contributing to the field by providing a standardized methodology for benchmarking cloud based applications. For determining the performance of web systems scalability, Deep Manishkumar Dave and Amit Bhanushali [5] presented a performance testing methodology. Stress testing and endurance testing is what the authors have measured application performance on prolonged heavy loads. Based on this work they obtained practice insights regarding how to reach thresholds of application scalability and application reliability under changing load conditions.

A. Comparison of Traditional vs. Modern Performance Testing Approaches

Existing traditional performance testing methods based on manual execution and predefined test cases were unable to cover the scalability assessment in an opti-mal manner. A new era of scalability testing and cloud benchmarking with automated method brings richness of accuracy and flexibility. Table I below highlights key differences:

Table I : Comparidon of Traditional and Modern Performance Testing Techniques

Feature	Traditional Testing	Modern Testing
Execution	Manual	Automated
Scalability Assessment	Limited	Comprehensive
Toolset	LoadRunner, JMeter	Gatling, K6, BlazeMeter
Real-World Simulation	Limited	Advanced AI-based models

B. Challenges in Performance Testing for Scalability

While a lot has been achieved in regard to the scalability testing, there still remain some challenges:

- Existing tools are unable to capture real world user behavior and traffic varieties and therefore they are inaccurate in their assessment of scalability in dynamic application environments.

- Most methods are not automated and require a lot of manual work time, which means that scalability of the testing process is difficult, error prone, and hard to repeat on large boundary cases.
- In highly distributed and cloud-native environments, performance testing is still an open issue since most of the resource orchestration complexities and workload fluctuations are not predictable.
- Performance inconsistencies are caused by dynamic workload variations and unpredictable traffic spikes, thus it becomes hard to attain stable scalability under a wide range of applications.
- It is difficult to integrate performance testing with DevOps pipelines, which needs seamless automation of performance testing, continuous monitoring and adaptive load testing that scale with the development of the system.

The focus of this paper is to bridge these gaps by proposing an automated, real time scalable load and performance testing framework with an objective of building an optimal test framework deploying a combination of automation, real time scalability analysis, and adaptive benchmark tool. This research presents a novel methodology which improves scalability testing accurateness and feasibility while contributing to the robust performance of the application in dynamic environments. In this paper, the rest of the text is structured as follows: first, section 2 discusses literature on the previous researches of the performance testing methodologies, scalability assessment techniques and the testing tools. In section 3, the proposed performance testing methodology as well as the framework, tools, and the evaluation criteria are outlined. Section 4 presents experimental comparison, validation of the proposed approach which is followed by Section 3, and presented in the experimental results. In section 5, the study's key findings and directions of inquiry are discussed, its implications and limitations. In the last section, Section 6 concludes the paper and sketches future directions research.

III. LOAD AND PERFORMANCE TESTING: CONCEPTS AND TECHNIQUES

It is necessary to test a performance of the application to guarantee its ability to bear different load levels and to remain stable and efficient. The topics discussed in this section focus on key concepts, tools and methodologies used for relevant to load and performance testing.

A. Load Testing

Load testing is used to evaluate an application's performance under expected and peak user loads [12]. Response time, throughput and resource utilization are assessed and the possible forks are also identified. Its primary goal is to guarantee system stability, detect performance degrading point, resource allocation optimization, SLA validation, etc, to improve user experience by minimizing the failure rate. The applications that vary in workload must go through load testing. Flash sales impose huge traffic surges on e-commerce platforms; likewise, banking applications need robust performance for handling concurrent transactions. Load testing is vital for cloud based services because it is dependent on its scalability under a dynamic workload. There are variety load testing tools available to make it effective. Apache JMeter is a tool for simulating user requests and a performance analysis. Enterprise level load testing of cross protocols is available in LoadRunner and continuous load testing with real time monitoring is enabled by Gatling. These tools aid in optimizing the application performance and preserving it in consistent manner under high user loads.

B. Performance Testing

Test of performance is the testing of the speed, responsiveness and stability of the system at different levels of workloads. But, it guarantees that an app brings up enough performance requirements right before the app gets deployed.

a) *Key Metrics*

i) *Response Time:*

The response time of a system is the time it takes for the system to process and respond to a user request. High response time is critical performance metric, as it has direct impact on the user experience, so when response time in an application is high, it results in an application sluggishness and dissatisfaction. Response time tuning helps maximize the interactions, and minimize the time for service efficiency.

ii) *Throughput:*

The throughput measures the number of transactions a system can handle per second, therefore study the overall system efficiency. In high traffic applications such as e-commerce and financial systems, the performance is said to be better when throughput is higher. Throughput monitoring helps organizations to identify performance bottlenecks and increases in transaction handling capacity.

iii) CPU & Memory Usage:

It is a mode of calculating the consumption of system resources, such as power of processor and memory, in the case of different workload conditions. Slow performance, crashes, or failures have the chance to occur due to excessive cpu or memory usage. Resource utilization optimization leads to an application stability due to the prevention of application performance degradation under its heavy user loads.

b) Types of Performance Testing

i) Stress Testing:

This is a testing that determines how a system would perform when it's pushed above its normal operating capacity. This would help identify breaking points, formulating failure recovery mechanisms, and finalizing system stability if the system works under excessive workload. Essential to such applications is stress testing to allow them to gracefully degrade rather than abruptly going down [13].

ii) Spike Testing:

This test is used to test the kind of an application that will handle sudden and unpredictable surges of user traffic. It examines if the system can quickly scale up resources within minutes or even seconds, and whether it remains stable and does not crash. More specifically, as in the case of ticket booking systems, deadlines on user traffic may be very short, but user traffic may ramp up quickly [11].

iii) Endurance Testing:

It is also known as soaking testing; this is a method wherein we evaluate a system's performance at continuous operation with sustained load for a long duration. It is able to find memory leaks, kill performance, and early signs of crashes. Applications that need to be continuously running require endurance testing [14].

iv) Tools and Frameworks:

K6 is an open source and modern performance testing tool built for integrating with CI/CD to help you conduct automated and continuous performance checks. BlazeMeter is a cloud performance testing platform for large scale tests performed with real time analytics to allow teams to do efficient evaluations of system performance under different conditions. Advanced capabilities offered to organizations by these tools for optimizing the performance testing strategy [15][16].

C. Role of Performance Testing in Scalability

Performance testing helps in finding out and removing scalability problems as it simulates real world loads by forcing the server to take a long time to respond under such high traffic. It will help find resource allocation inefficiency, latency of the network, and database performance in order to optimize system performance. Furthermore, performance testing ensures that the cloud based application can adapt and scale dynamically to meet the fluctuating traffic loads without having to worry performance degradation and offering a smooth user experience. Through systematic evaluation of these factors, the organizations are able to preemptively deal with scalability challenges and deriving robust software architecting. Industry best practices for an effective scalability dictate that all performance tests should be integrated into the software development life cycle early in the development phase in order to avoid costly late stage performance bulges and fixes [17]. In the DevOps pipelines, tests should be automated in order to find out the scalability issues early and do continuous system optimization. The use of real-world traffic patterns and load distributions improve the accuracy of the performance assessment and therefore increase the reliability of the test. System performance can be compared against what is considered the industry standards and competing companies to effectively optimize resource utilization and direct infrastructure planning. Following these methodologies and best practices will imply that organizations have ensured that their applications are nevertheless scalable, effective, and durable at managing dynamic workloads [18].

IV. EXPERIMENTAL DESIGN AND TESTING METHODOLOGY

In this section, experimental setup and methodology for performance testing are commented as it provides a systematized evaluation of the system scanning and efficiency. It covers from system architecture, testing framework, performance metrics, benchmarking criteria and a load testing framework.

A. System Architecture for Testing

The system architecture for performance testing is composed of interconnected multiple components which can simulate real world application environment. The architecture in Fig. 1 depicts a client server model in which users send their requests through a web or mobile interface until the requests are processed by the application server, having interaction with a backend

database to retrieve and store the data [19]. A big part of this setup is evaluating how the server clients interact with each other, request processing efficiency, how the responses are generated and how load balancing does in dispersing traffic. In addition, there are database load considerations involved in analyzing query execution efficiency, handling transactions and response times so that the database performance is not degraded in the presence of high concurrent user load [20].

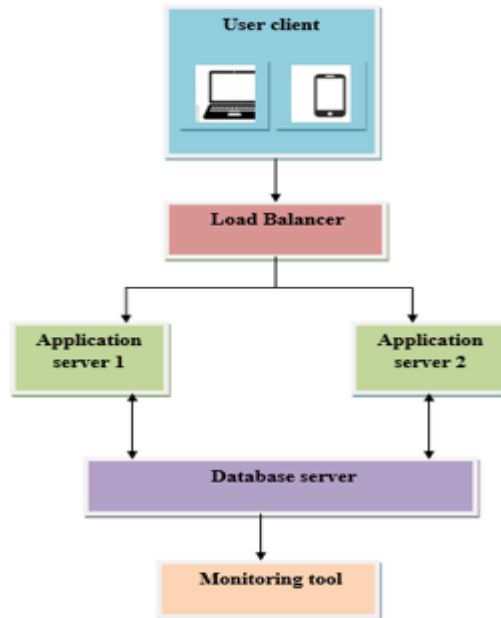


Figure 1 : System Architecture Diagram for Performance Testing Setup.

B. Performance Testing Framework Implementation

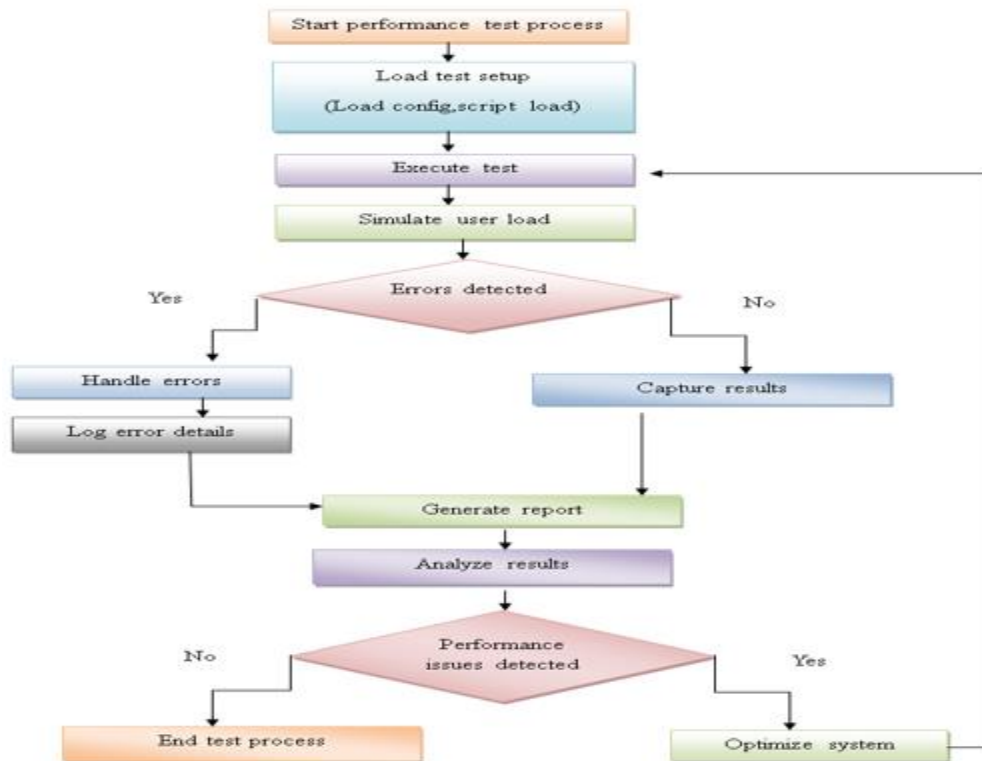


Figure 2 : Flowchart of Automated Performance Testing Process.

The framework of the performance testing follows a well structured workflow to evaluate system behavior at different loads to obtain thorough assessment and optimization. Testing workflow is graphically presented in a flowchart form, where, among other phase, test initialization, script execution, workload simulation, result collection, and analysis are specified. Performance tests are scripted using Apache JMeter, LoadRunner, or Gatling using tool where scripts define user’s actions, concurrent requests and data payloads and mimic actual traffic patterns. The framework is made to be integrated with the CI/CD pipelines to run performance test automatically in different phases of the software development lifecycle. This automation guarantees the early detection of the performance issues, continuous monitoring and proactive the scaling. As per this structured approach, organizations can eliminate performance evaluations issues, optimize resources utilization and address scalability constraints in an effective manner. Fig. 2 illustrates the flowchart of Automated Performance Testing Process.

C. Metrics and Benchmarking

Key performance indicators (KPIs) and benchmarking criteria are set to quantitatively assess system efficiency and compare results against industry standards so as to provide the system performance. A critical KPI when it comes to response time is the time it takes for the system to process user request and send back response to the user without having any issues that cause bad user experience. Another important metric is throughput that is the number of successful transactions processed per second, which represents the scale of a system when serving concurrent request. In addition, CPU and memory utilization are studied for different workload conditions to achieve the best performance without unreasonably utilising the resources. The expected and peak workloads are simulated, and against industry standards from e-commerce, banking and cloud computing. This provides a method to validate the scalability of the system, meaning that performance is fit for the real world application.

D. Pseudocode for Automated Load Testing

Performance tests at scale require performing automated load testing. The next piece of pseudocode describes a simple automated test execution process:

Table 2 : Pseudocode for Automated Load Testing

Pseudocode for Automated Load Testing	
Initialize_Test_environment ()	# step 1: Initialization
Configure_Test(user_load, request_rate, duration)	# step 2: Load test configuration parameters
For each virtual_user:	# step 3: Simulate virtual users
Begin	
Start_session()	
Send_request_to_server()	
Capture_response_time()	
Capture_throughput()	
Log_results()	
End_session()	
End	
End for	
Aggregate_test_results()	# step 4: Aggregate test results
Generate_performance_report()	# step 5: Generate performance report
Analyze_performance_issues()	# step 6: Analyze Performance Issues and Optimization Strategies
Recommend_optimization_strategies()	

Automated performance testing tools comprise of have a core functionality shown here in this pseudocode. Automated tests perform well, as they allow executing multiple virtual user sessions to measure response times, collecting results and providing nice insights on system behavior when under varying load conditions.

V. Experimental Results and Analysis

This section presents the findings from the performance tests, including test execution outcomes, scalability assessments, comparative evaluations of testing tools, and identified challenges. The results are analyzed using key performance metrics and are supported by visual representations such as graphs and tables to provide comprehensive insights into the system’s behavior under different workloads.

A. Test Case Execution and Observations

System responsiveness, stability and resources consumption were evaluated under varying user’s loads through performance tests. A goal was to determine how the system performs under various levels of concurrent user requests to identify possible performance bottlenecks as well as system constraints.

a) Performance Testing Results

Under these three primary load conditions, three primary load conditions (light load - 10-50 concurrent users, moderate load - 50 - 200 concurrent users, peak load - 200 - 1000 plus concurrent users) the performance testing was conducted. The varying user loads were used to measure key performance indicators (KPIs) that include response time, CPU utilization, memory consumption, etc. The data was analyzed completely aggregated. The potential performance bottlenecks were identified by evaluating which of these metrics be able to sustain the performance of the system on various workload conditions. The results are shown with the following graphs:

b) Response Time vs. Concurrent Users

One of the observed key performance metrics was the response time, the amount of time the system takes to process user request. This fig. 3 shows that the response time is vary depending the number of concurrent users. Usually, as the concurrency level increases, the latency also increases, implying that the system may be running into bottlenecks.

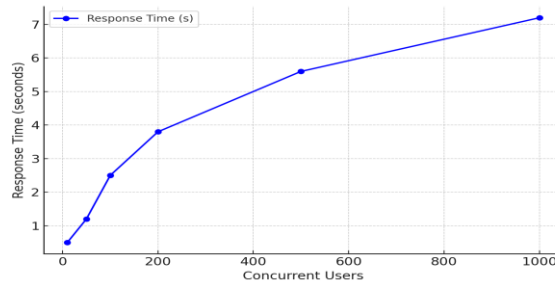


Figure 3: Response Time VS. Concurrent User Graph.

c) CPU and Memory Utilization under Load:

The second performance metric measured was resource utilization (memory and CPU consumed in the system). On the other hand, the fig. 4 shows CPU and memory usage variations depending on users’ load and critical points where the system performance will suffer from high resource consumption. To have a structured view of how performance varies under different cases, the following table presents the CPU and memory metrics utilization under different test conditions.

Table 3 : CPU And Memory Utilization Under Increasing Load

Number of Concurrent Users	Response Time (ms)	CPU Utilization (%)	Memory Usage (MB)	Error Rate (%)
10	120	15	250	0.1
50	200	30	500	0.5
100	350	45	750	1.2
200	500	60	1100	2.5
500	950	80	1800	5.0
1000	1600	95	2500	8.3

Table 3 shows the results indicate that response time, CPU utilization and memory consumption are significantly high with an increase in the number of concurrent users. Depending on the system saturation, the error rate increased at peak loads.

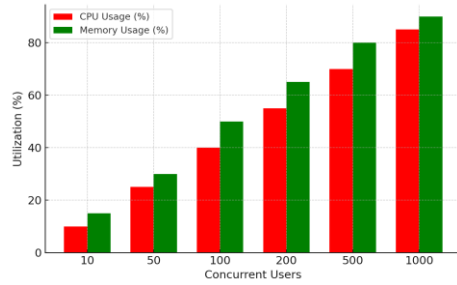


Figure 4 : CPU And Memory Utilization Under Increasing Load Graph

B. Scalability Assessment

The scalability is testing how the system can accommodate increasing user volumes in a concurrent way with minimal degradation of the performance. It evaluates the system response to workloads change and resource allocation under peak conditions for the systems. By observing the trends, one can get valuable insight in the way the key performance metric changes with the scale of the work load. It analyzes whether performance degrades linearly or exponentially (or more) or whether it stays the same under stress. Application responsiveness and server resource distribution are also studied with respect to concurrent users so that we can identify which points in the user space lead to bottlenecks. Understanding these trends, system architects should be able to make strategic optimizations with running systems, enhance infrastructure, allocate resources and make application scaling stronger.

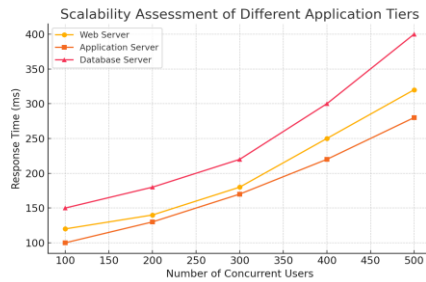


Figure 5 : Scalability Assessment of Different Application Tiers Graph.

C. Comparative Analysis

A comparative analysis on the usefulness of application/performance test tools such as Apache JMeter, LoadRunner, Gatling, k6 was conducted to evaluate their efficiency. Then each tool was rated in terms of ease of scripting, execution speed, scalability and fit for integration with CI/CD pipelines. The parameters such as automation capability, Distributed load generation, Accuracy in measuring KPI were analyzed. The findings provide direction to most suitable tool for performance testing in early stages of a system’s life cycle for scalability and reliability.

Table 4 : Performance Comparison of Load Testing Tools

Feature/Tool	Apache JMeter	LoadRunner	Gatling	k6
Ease of Scripting	Moderate	High	High	Moderate
Distributed Load Generation	Yes	Yes	Yes	Yes
CI/CD Integration	Moderate	High	Moderate	High
KPI Measurement Accuracy	High	High	Moderate	High
Execution Speed	Moderate	High	High	High
Open-Source	Yes	No	Yes	Yes

Table 4 comparative assessment helps in deciding which the best is tool regarding the project requirements, scalability and automation requirements.

D. Limitations and Challenges

Although there has been structured performance testing, some of the real world constraints and challenges were encountered. Currently, simulating traffic on large scale with realistic user behavior is complex because hardware and network limitations can influence the test accuracy. Moreover, the results of the test are also influenced by the internet speed, device configurations and geographic locations variability, making it hard to achieve results consistently. Performance evaluations further become complicated due to accurately modelling the dynamic interactions of users such as caching, API throttling and session handling. These challenges need to be overcome through a careful test design over reliable infrastructure and advanced traffic simulation techniques as part of a realistic and reliable performance assessment.

VI. CONCLUSION

The load and performance testing is highlighted in this research to assess application scalability. The study shows how various the testing methodologies and tools can be used to determine system bottlenecks and allocation of resources. Experimental results show that as concurrently loads increase, response times and resource utilization also increase dramatically, which decreases stability of the application. Performance testing tools are compared to determine why modern performance assessment uses automation and CI/CD integration. Performance testing holds value, but the real world traffics and dynamic workloads still need to be simulated. Future research should aim at increasing the automation and the real time performance monitoring, as well as the adaptive benchmarking techniques for scalability testing frameworks. By implementing the robust performance testing strategies, applications will always be able to have high availability, responsiveness, and reliability in dynamic environments.

VII. REFERENCES

- [1] A. Avritzer, R. Britto, C. Trubiani, M. Camilli, A. Janes, B. Russo, and R. K. Chalawadi, (2022). Scalability testing automation using multivariate characterization and detection of software performance antipatterns. *Journal of Systems and Software*, 193, 111446.
- [2] F. Gortázar, M. Gallego, M. Maes-Bermejo, I. Chicano-Capelo, and C. Santos, "Cost-effective load testing of WebRTC applications," *Journal of Systems and Software*, vol. 193, pp. 111439, 2022.
- [3] N. Mungoli, "Scalable, distributed AI frameworks: leveraging cloud computing for enhanced deep learning performance and efficiency," 2023. arXiv preprint arXiv:2304.13738.
- [4] S. Henning, and W. Hasselbring, "A configurable method for benchmarking scalability of cloud-native applications," *Empirical Software Engineering*, vol. 27, no. 6, pp. 143, 2022.
- [5] D. M. Dave, and A. Bhanushali, "Performance Testing: Methodology for Determining Scalability of Web Systems".
- [6] G. Blinowski, A. Ojdowska, and A. Przybyłek, "Monolithic vs. microservice architecture: A performance and scalability evaluation," *IEEE access*, vol. 10, pp. 20357-20374, 2022.
- [7] M. A. H. Emu, M. Mahmood, M. M. Asif, A. R. Tanvir, and R. S. Joyeeta, "Designing a new scalable load test system for distributed environment (Doctoral dissertation, Brac University)," 2022.
- [8] S. Pargaonkar, "A comprehensive review of performance testing methodologies and best practices: software quality engineering," *International Journal of Science and Research (IJSR)*, vol. 12, no. 8, pp. 2008-2014, 2023.
- [9] S. Chinamanagonda, "Cloud-native Databases: Performance and Scalability-Adoption of cloud-native databases for improved performance," *Advances in Computer Sciences*, vol. 6, no. 1, 2023.
- [10] P. Loncar, and P. Loncar, "Scalable management of heterogeneous cloud resources based on evolution strategies algorithm," *IEEE access*, vol. 10, pp. 68778-68791, 2022.
- [11] A. C. Barus, E. S. Sinambela, I. Purba, J. Simatupang, M. Marpaung, and N. Pandjaitan, "Performance Testing and Optimization of DiTenun Website," *Journal of Applied Science, Engineering, Technology, and Education*, vol. 4, no. 1, pp. 45-54, 2022.
- [12] M. Yenugula, R. Kodam, and D. He, "Performance and load testing: Tools and challenges," *International Journal of Engineering in Computer Science*, vol. 1, pp. 57-62, 2019.
- [13] M. Hendayun, A. Ginanjar, and Y. Ihsan, "Analysis of application performance testing using load testing and stress testing methods in API service," *Jurnal Sisfotek Global*, vol. 13, no. 1, pp. 28-34, 2023.
- [14] A. Kovács, G. Á. Németh, P. Sótér, "INTEGRATING PERFORMANCE TESTING INTO CONTINUOUS INTEGRATION LOOPS," *In Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae. Sectio Computatorica*, vol. 54, 2023, January.
- [15] M. Shushakova, "Improving software quality through non-functional testing (Master's thesis, M. Shushakova)," 2023.
- [16] B. Zibitsker, and A. Podelko, "Performance Testing and Modeling for New Analytic Applications," *Big Data and Advanced Analytics*, no. 6-1, pp. 19-32, 2020.
- [17] Y. Yang, G. Kissas, and P. Perdikaris, "Scalable uncertainty quantification for deep operator networks using randomized priors," *Computer Methods in Applied Mechanics and Engineering*, vol. 399, pp. 115399, 2022.
- [18] S. Zeng, S. Pian, M. Su, Z. Wang, M. Wu, X. Liu, and G. Tao, "Hierarchical-morphology metafabric for scalable passive daytime radiative cooling," *Science*, vol. 373, no. 6555, pp. 692-696, 2021.

- [19] L. Zhang, J. Zhao, P. Long, L. Wang, L. Qian, F. Lu, and D. Manocha, "An autonomous excavator system for material loading tasks. *Science Robotics*," vol. 6, no. 55, pp. eabc3164, 2021.
- [20] M. G. Khan, N. U. Huda, and U. K. U. Zaman, "Smart warehouse management system: Architecture, real-time implementation and prototype design," *Machines*, vol. 10, no. 2, pp. 150, 2022.