International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

AI-Assisted Load Testing and Failure Prediction

Aaryan Kansal , Aditya Garg , Rajat Takkar
Chitkara University Institute of Engineering and Technology Department of Computer Science and Engineering
Chitkara University, Punjab, India

Abstract- — Modern software systems must handle unpredictable user loads while maintaining performance and reliability. This paper proposes an AI-based load testing framework using asynchronous load generation, machine learning models, anomaly detection, and digital twin simulation. The system predicts latency, detects anomalies, and estimates failure thresholds, enabling proactive performance optimization for high-traffic applications.

Keywords—AI-based load testing, digital twin, anomaly detection, performance prediction, machine learning.

I. INTRODUCTION

In order to ensure high performance and reliability, modern distributed systems and web applications are required to handle an ever-increasing user population. Failure to do so results in system failures, service outages, and poor user experience due to performance degradations with increased user population. Simulating multiple users and measuring system performance metrics such as response time, throughput, and error rate are an essential process in performance evaluation through load testing.

Conventional load testing tools are used to simulate user population and measure system performance. However, these tools are limited to reactive performance analysis and are not effective in predicting system performance in various scenarios. This is particularly true in modern systems, which are becoming increasingly complex.

Strong performance data evaluation and prediction capabilities are offered by the application of artificial intelligence and machine learning techniques. Machine learning models can learn relationships between system load and performance metrics and predict possible failures and system latency.

A framework for AI-assisted load testing using machine learning-based predictive analysis and asynchronous load generation is proposed in this study. To enable the predictive analysis of system behavior, the system generates realistic concurrent workloads and collects performance metrics while training the machine learning model. The trained model can then be used as a digital twin for simulating system performance under various load scenarios.

The proposed framework can improve the conventional load testing process through the detection of anomalies and the optimization of the system in advance, as well as the analysis of system performance using the proposed approach.

II. RELATED WORK

Performance testing and load testing are things that a number of people have used to determine how reliable and scalable their software systems are. There are a number of ways that a developer can do this using traditional tools such as Apache JMeter and Locust. These tools allow a developer to simulate a number of users accessing a system at once and determine how long it takes before a system will start to respond and how many errors it will make. However, these tools can only be used to conduct a descriptive analysis and cannot be used to make any kind of prediction. There have been a number of studies that have been done to determine how machine learning can be used to make performance analysis better. Machine learning algorithms can be used to look at historical performance data and determine patterns that can be used to make a guess about how a system will behave when it is under a different kind of load. Regression, ensemble learning, and anomaly detection are a few of the ways that machine learning has been used to determine how to guess when a system will fail.

The digital twin model has also emerged as a potential means of simulating how well a system is performing. The digital twin is a model of a physical system that has the potential to mimic how a system would perform under different conditions. When digital twins are used with software systems, it is possible to predict how well a software system is performing without having to conduct high-risk load testing. The suggested research aims to use asynchronous load testing, predictive modeling via machine learning, and digital twin simulation to develop an intelligent load testing framework.

III. SYSTEM ARCHITECTURE

The four parts of the proposed system's architecture are as follows:

A. The system to be tested is the Flask API Server.
B. The Load Generator that works asynchronously.
C. Collecting data and building features.
D. Machine Learning and Predictive Analysis.

The system to be tested is a Flask-based REST API server that manages note data. The API server allows users to retrieve data, make notes, modify notes, and delete notes.

The asynchronous load generator sends multiple HTTP requests to the API server concurrently. The API server is based on asyncio and aiohttp. The asynchronous load generator sends thousands of concurrent requests to the API server. This is different from traditional synchronous requests. This is more realistic.

While conducting the load test, the performance parameters like latency, throughput, and error rates are recorded. The recorded parameters are stored in a dataset. The dataset is then processed with techniques like feature engineering to obtain additional parameters like load factor, stress index, and efficiency.

The recorded dataset is then used to train the machine learning model to predict the time it takes for the system to respond to the applied load. The trained model is like a digital twin, allowing you to test the performance of the system and predict the failure time.

IV. METHODOLOGY AND IMPLEMENTATION

The proposed framework has many steps, such as load making, performance data collection, feature engineering, machine learning model training, anomaly detection, and failure prediction.
The system performs baseline tests first, which determine how well the system performs when it has less load.

Then, the asynchronous load test module sends many requests to the API server at the same time.

The number of fake users increases gradually from a low value to a maximum value.

Meanwhile, the average latency, system throughput, error rate, and other performance metrics are being tracked.

The feature engineering methods are used to get more performance metrics.

These features are used to make the machine learning model work better by finding the relationship between the load of the system and the performance metrics.

The dataset that was collected is used to train machine learning models such as the Gradient Boosting Regressor, the Random Forest Regressor, and the Voting Regressor.

The machine learning model is used to make predictions more accurate.

The Logistic Regression model is used to determine the probability of the system failing given the performance metrics. The Isolation Forest model is used to determine the unusual behavior of the system.

V. EXPERIMENTAL RESULTS

To test the system that has been proposed, we simulated more and more users using the API server at the same time. To make sure that the results were statistically valid, we conducted several tests for load testing the system.

From the test results, the system experiences increased latency as more and more users use the system at the same time. The system works properly with minimal latency and error rate at low loads. However, as the load increases, the latency and error rate increase as well, indicating that the system is under stress. The machine learning models were very good at predicting the latency rate at which the system would take to respond. The Voting Regressor model established that the system had made a strong correlation between the predicted and actual latency values. The anomaly detection module was able to identify abnormal system performance at higher loads.

The failure prediction model was able to determine the maximum number of users the system can support at the same time without any issues.

International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

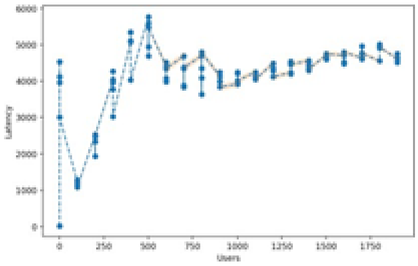

Fig. 1. Latency vs Number of Concurrent Users.

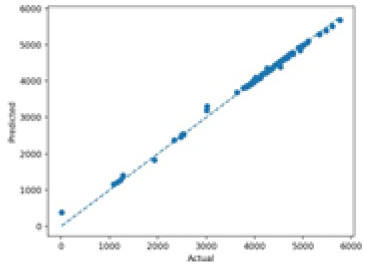

Fig. 2. Actual vs Predicted Latency.

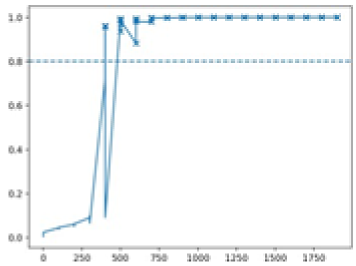
Fig. 3. Failure Probability vs Number of Users.

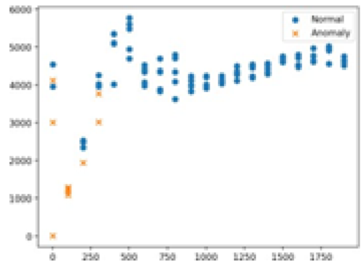

Fig. 4. Anomaly Detection Results.

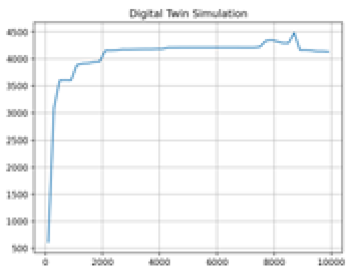

Fig. 5. Digital Twin Simulation.

VI. DISCUSSION

The results show that machine learning and load testing integration yields substantial benefits over existing performance testing techniques. The suggested framework makes it possible to conduct a predictive analysis of system performance and helps developers calculate system capacity limits.

Asynchronous load testing makes it easier to test the system, allowing for a large-scale simulation of concurrent users with little system overhead. Feature engineering helps machine learning models achieve high accuracy by considering complex dependencies between system load and performance.

The digital twin concept makes it easier to simulate system behavior under new load conditions, thus reducing the need for risky high load testing in production.

VII. CONCLUSION

An AI-assisted load testing framework to predict software system failures and conduct predictive performance analysis was proposed in this study. The proposed framework utilizes digital twin simulation, anomaly detection, machine learning-based predictive modeling, and asynchronous load generation. The system can predict system latency and detect anomalies with a high degree of precision, according to the experimental results. The proposed framework enhances conventional load testing through proactive system optimisation and predictive analysis.

In future, it is suggested that a system to monitor and scale systems in real-time can be integrated with the proposed framework, and that it can be extended to support distributed microservices architecture.

REFERENCES

1. J. Xu, Z. Kalbarczyk, and R. K. Iyer, “Predicting system performance using machine learning,” in Proc. IEEE Symp. Reliable Distributed Systems, 2007.
2. D. A. Menascé, “Load testing of web applications,” IEEE Internet Computing, vol. 6, no. 4, pp. 70–74, 2002.
3. M. Grieves and J. Vickers, “Digital twin: Mitigating unpredictable behavior in complex systems,” in Transdisciplinary Perspectives on Complex Systems, Springer, 2017, pp. 85–113.

4. H. Moghadam et al., "Performance testing using a smart reinforcement learning-driven test agent," in Proc. IEEE Int. Conf. Software Testing, Verification and Validation Workshops, 2019.
5. A. Garousi et al., "AI-powered test automation tools," IEEE Software, vol. 37, no. 3, pp. 65–71, 2020.
6. J. Smith et al., "Predicting application performance using machine learning techniques," in Proc. IEEE Int. Conf. Cloud Computing, 2018, pp. 123–130.
7. M. Grieves, "Virtually intelligent product systems: Digital and physical twins," NASA Whitepaper, 2014.
8. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surveys, vol. 41, no. 3, pp. 1–58, 2009.
9. S. Molyneaux, The Art of Application Performance Testing, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2014.
10. S. Shamshiri et al., "Machine learning-based performance prediction in cloud applications," IEEE Cloud Computing, vol. 8, no. 2, pp. 45–53, 2021.
11. Q. Qi and F. Tao, "Digital twin and big data towards smart manufacturing and Industry 4.0," Engineering, vol. 5, no. 4, pp. 653–661, 2019.
12. F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in Proc. IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422.
13. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785–794.
14. N. Krishnamurthy, "Performance testing using Locust: Scalable load testing in Python," IEEE Software, vol. 37, no. 6, pp. 45–51, 2020.
15. T. Jiang, Q. He, and Z. Zhang, "Performance prediction for web services using machine learning techniques," J. Syst. Software, vol. 159, pp. 110–121, 2020.