Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi
Sayı 8, S. 109-117, 2, 2024

Araştırma Makalesi

International Journal of Advanced Natural Sciences and Engineering Researches
Volume 8, pp. 109-117, 2, 2024

Research Article

https://as-proceeding.com/index.php/ijanser
ISSN: 2980-0811

Comparative Analysis of AI-Supported and Manual JMeter Tests: The Role of Generative AI and LLM in Software Performance Testing

Burak TUZLUTAŞ[1], Murat ŞİMŞEK [2]

[1]*Software Engineering / Institute of Science, Ostim Technical University, Türkiye*
[2]*Artificial Intelligence Engineering / Institute of Science, Ostim Technical University, Türkiye*

Email of corresponding author: buraktuzlutas@gmail.com
Email of corresponding author: murat.simsek@ostimteknik.edu.tr

(Received: 20 February 2024, Accepted: 08 March 2024)

(4th International Artificial Intelligence and Data Science Congress ICADA 2024, March 14-15, 2024)

ATIF/REFERENCE: Tuzlutaş, B. & Şimşek, M. (2024). Comparative Analysis of AI-Supported and Manual JMeter Tests: The Role of Generative AI and LLM in Software Performance Testing. *International Journal of Advanced Natural Sciences and Engineering Researches, 8(2), 109-117.*

Abstract **-** This paper addresses the challenges of conducting software performance testing and the challenges encountered in the pre-testing process. The focus is on the importance of software performance testing and evaluation methodologies. At the same time, the main theme of large language models (LLM) and the characteristics of modeling and its role in this process are examined.
The overall aim of the study is to investigate how Generative AI- Large Language Models (LLM) can be used efficiently in performance testing in important stages such as creating test plans, constructing test profiles, creating and preparing data, and interpreting the reports received as a result of the tests. The advantages of Artificial Intelligence, more precisely Generative AI- Large Language Models (LLM), are discussed in terms of optimizing the processes carried out in performance testing in a positive sense and accelerating the process.
This study is envisioned as a contribution to the traditional methods used in performance testing. The potential of Generative AI-Large Language Models (LLM) to effectively solve the problems in traditional testing methods and to create more efficient testing processes may guide the development of performance testing methodologies in the future.

Keywords: Software Performance Testing, Artificial Intelligence, Generative Aı, Large Language Models, Software Testing

I. INTRODUCTION

Software performance testing is a testing process to evaluate how an application, system or a software module performs and how it works under certain conditions. As a result of the tests, it analyzes how the system works and behaves and analyzes response times, memory usage, processor performance and different related criteria[1].

Although the purpose of performance testing is generally focused on performance metrics, it is also performed to detect potential problems and evaluate its scalability by focusing on how it will perform under certain load. These tests are performed to push the application to its limits and as a result, to determine its limit so that it can run more reliably, quickly and efficiently.

This work is envisioned to be a contribution to the traditional methods used in performance testing. The potential of Generative AI-Large Language Models (LLM) to effectively solve the problems of traditional testing methods and to create more efficient testing processes may guide the development of performance testing methodologies in the future.

Performance testing is generally focused on performance metrics, but it can also be used to identify potential problems and evaluate scalability by focusing on how the application will perform under a given load. These tests are performed to push the application to its limits and ultimately to determine its limit so that it can run more reliably, quickly and efficiently.

There are different types of tests to perform performance testing, usually using different profiles such as load tests, endurance tests, stress tests, scalability tests and speed tests. During these tests, different performance criteria of the system are measured and scenario-based analysis is aimed on the reports generated.

Fig. 1 Working Principle of the Structure

Test Types

Load Testing : It is the type of test performed to determine how the system will behave under a certain number of users and to determine the maximum user-transaction volume.

Stress Testing : It is a test performed to determine how the application, system or a software module behaves when loaded above the normal load profile by going beyond its limits. It is aimed to detect performance problems in unexpected load scenarios in advance[2].

Endurance Testing : It is a type of test that is performed for a long time rather than short-term tests to analyze problems such as memory overruns and resource consumption. These tests are endurance tests that are usually performed to detect errors that are not visible during short-term tests and are periodically received during certain test periods.

Performance Profiling : Analyzes performance profiles between different components and functions of the software to determine where and how much resources are consumed.

Configuration Testing : It is a type of performance testing in which the performance of the software is tested with configuration changes made on hardware and software[4].

Important topics to be determined before performance tests;
Data Collection: Using the necessary data during the tests and determining the test data in the processes.

Creating a Test Plan: Determination of test cases, objectives and methods. Determining the thread structure and test methods for the end to end process or endpoints expected to be tested.

Determining Test Profiles: Determining which test profile will be used in the prepared scenarios.(Stress test, Load test)

Interpretation of Reports: Taking the necessary actions as a result of the test plans prepared and the test profiles determined and the analysis reports received.

Softtech and Products

Softtech, a subsidiary of Isbank, is a leading company in the software industry. It produces software solutions in various sectors such as finance, human resources and e-commerce. It offers modular and scalable software solutions with a high number of customers.

The Importance of Performance Testing for Softtech

For Softtech, performance testing is to ensure speed, reliability and scalability in the software solutions produced and in the development process. As a result of these tests, it is carried out in order to optimize the products and ensure software stability. The operation of this process in all modules that have been developed or continue to be developed in the tests is very critical in terms of performance optimization in systems under intensive use. Since it is aimed to predict how all software products developed within Softtech will behave in intensive use and to take precautions, many different tests are performed depending on performance profiles.

Problems During Performance Testing Experience at Softtech

The uncertainties that occur during the preparation of the test plan, the use of incorrect or incomplete data during the data creation or collection process, the inaccurate determination of the test profiles to be focused on, and the reports received as a result of these uncertainties cause incomplete and far from real results. Failure to correctly determine these metrics negatively affects the effectiveness of the performance tests and causes the improvements envisaged in the software to be inaccurate.

In the light of the information provided, it is aimed to address the main problems encountered during the performance tests performed at Softtech. Without focusing on the performance testing processes, questions such as at which stage of the process there is slowness, at which stages of the process the problems actually slow down the process before the test, and which strategy will be used to solve them are emphasized.

Large Language Models (LLM) and Performance Testing:

Generative AI and Large Language Models (LLM) are aimed to provide the best optimization in performance testing processes by using them effectively in stages such as determining test profiles, data generation, test plan generation and report interpretation in software performance testing. Creating the scenario and determining the test profile is the most important stage in performance testing processes. Since failure to make the right setup means that the outputs of the test will not produce the desired results, it is of great importance to operate the structure correctly here. If the correct analysis is made on the raw data and the generated report with Large Language Models (LLM) after the tests, efficient test outputs will be generated and correct inferences will be made on the performance criteria. It is predicted that the improvements made according to the feedback here will directly affect the performance in a positive way[3].

II. MATERIALS AND METHOD

We will conduct a feasibility study to create a jmx file to run the performance tests, and to interpret the results from jtl files.

It provides information about how Generative AI and Large Language Models (LLM) are used to interpret the reports and create test plans with the specified question sets. Generative AI will be used to create a new jmeter file based on the user's input.

Large Language Models (LLM) will be used to extract meaningful information from large amounts of textual data by making the jtl file, which jmeter provides raw before reporting, comprehensible or analytical.

Interpretation of Generated Reports in jMeter

The analysis of the visuals was carried out through "Chat -Ask Chatbot Assistant". The response received for the image below is "There is a distribution showing how the response times change in the 90th, 95th and 99th percentiles. While they are generally consistent at a low time, there are some peaks (high values). These peaks may indicate performance degradation under high load on the system." It is shaped as follows.

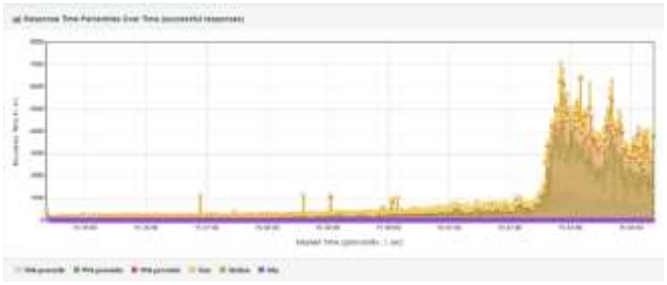

Fig. 2 Percentages of Response Time Over Time (successful responses)

In Figure 3 shows the increasing number of active threads during the test. This linear increase can be considered as the designed scenario of the load test. Towards the end of the test, the number of threads stabilizes and then decreases. This may signal the termination phase of the test.

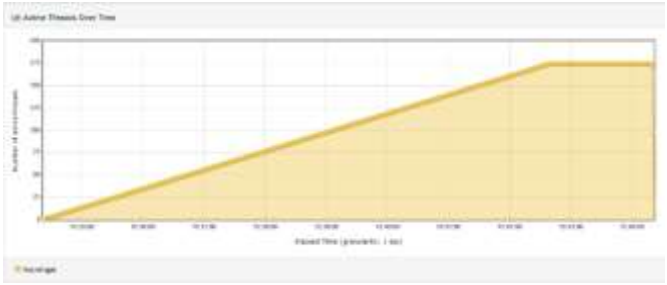

Fig. 3 Active Threads Over Time

It shows the average response times of the specific services tested (users and self-resources). Both services show a significant increase in response times towards the end of the test. This indicates that the services are performing poorly under load.

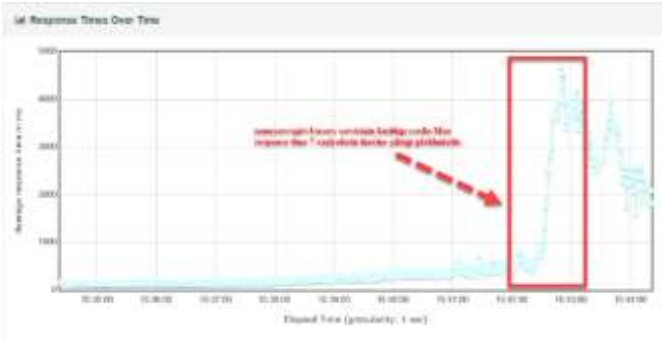

Fig. 4 users API Response Time Over Time

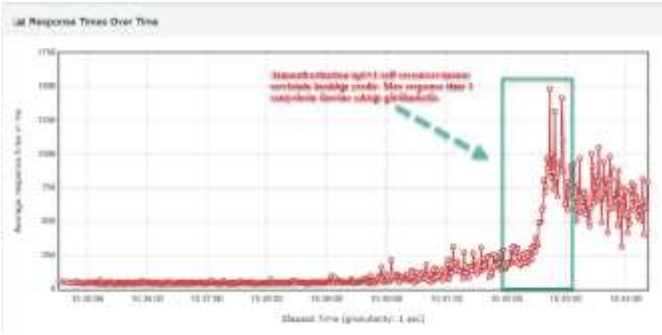

Fig. 5 self-resources API Response Time Over Time

In Figure 5, the CPU utilization and quota status shows that some services have high CPU utilization. In particular, the CPU usage of the 'gateway' service is quite high and is approaching its quota limit (89%). The CPU usage of the 'iamauthorization' service is very close to its quota (99.82%), which may indicate that this service is running at the limits of its resources.

Fig. 6 Grafana Monitoring

As a result of all this information, visuals were shared and asked to produce an output and a response was received as follows.

"In the light of this information, it can be said that during the performance test, some services experienced performance problems under high load and this may be related to the CPU utilization of the services. It is also possible that at a certain point in the test, the number of threads and response times increased and the system reached its limits under this load or experienced configuration problems. Such tests are critical for understanding system performance and identifying potential bottlenecks."

Preparing jmx Files to Run in Jmeter

For the test plan that we expect to be prepared on jmeter during the performance test study, we expect to generate results in specific test scenarios from simple to difficult with Generative AI thanks to the answers to the question sets given below. First, it is aimed to get information about the general test plan from the user. Here, information about the test profile to be performed is also obtained from the user.

1.General Test Plan Information

What is the name of the Test Plan?
With how many users is it planned to perform the test?(Thread)
What is the ramp-up period for users to start the tests?
How many times will the tests be performed?
What is the total duration of the test (Duration)

2. User and Data Source

How will the data needs for services be met (Direct - External CSV - Random Data Generation)?
What will be the variable names for service-specific user information?

3. HTTP Content

Which URLs do you want to test? Please also specify the HTTP method (GET, POST, etc.) for each one.
If you are using a method like POST or PUT, what data structure will be sent (e.g. JSON, form-data)
Do you want to add any HTTP Header? (Entering the parameters to be given in the Header according to the answer)

4. Reporting

Which type of reporting is desired (Graphical Results, Table View)?

The answers received within the framework of the above questions were not sufficient for a direct jmx file and were made meaningful by placing them in a code block within the prepared test plan. In the image below, only the addition of services and the CSV Data Set Config feature are used to make sense of receiving data from outside as a result of the answer given.

Fig. 7 XML File Received with Generative AI

III. RESULTS

The interpretation of the reports and the creation of the Jmx file were experienced during the interpretation of 5 reports and the creation of 7 different jmx files. As a result of the results obtained, for 2 topics ;

Interpretation of Reports

As a result of the studies, there were significant similarities between the outputs we normally obtain for the interpretation of images and the outputs obtained with GPT. The most important inference we observe here is the observation that the cost of time is reduced and that different outputs and observations are presented that we did not notice. In the outputs of the reports interpreted with AI, the inference of CPU-induced bottlenecking of the services extracted directly with our analysis was made directly with AI and the relationship between the visuals contributed positively to our evaluations.

By analyzing the contexts between Image 2, Image 3 and Image 4, the information about how many threads were broken in the Active Thread Over Time graph at the points where the services broke was pointed out with AI.

Creating the Jmx File

While creating the jmx file, a question directory consisting of 4 questions and sub-questions was prepared. A comparison was made between the resulting jmx file and a jmx file prepared by us through the Jmeter GUI. It was envisaged to answer the parts of the question set, which we divided into sub-questions, one by one by GUI and AI. The result of this is summarized in the table below.

Table 1 GUI vs Generative AI

Metrics	**Jmeter GUI**	**Generative AI**
Name of the Test Plan?	Success	Success
How many users will be tested?	Success	Success
Ramp-Up	Success	Success
How many times will the tests be performed?	Success	Success
Total time information of the test?	Success	Success
How will data needs be met?	Success	Configuration Missing
Configuration Missing Which URLs do you want to test?(Method name)	Success	Success
Do you want to add HTTP Header?	Success	Success
What type of reporting is required?	Success	Success
Direct execution of jmx in Jmeter can be added by editing the Xml format.	Success	It can be added by editing in xml format.

IV. DISCUSSION

Previously, studies on this construct have been done on the UI and API tests. However, performance tests are a type of test that is open to interpretation based on UI and API tests and requires analysis and report generation. Here, it is aimed to analyze the project that performs the test and the project where the tests are performed correctly and to decide to make the right performance improvements and / or improvements on the system side as a result of the outputs obtained in line with this information. Here, in

past studies, direct information about the improvements to be made by looking at the results was generally expected. The general purpose of our study is to analyze the metrics that may lead to performance losses that the user may miss as a result of time cost and analysis in an accurate and short time.

V. CONCLUSION

In this feasibility study, the interpretation of the reports and the creation of the jmx file with question sets are given below.

Interpretation of the Report;

Figure: 6 Based on the Grafana Tracking image, below is the response the AI received. It is aimed to analyze the point where the service shown in the red box in Figure 6 consumes 99% CPU.

"The CPU usage graph shows the CPU usage rates of various services. The "iamusers" service has low CPU usage, while other services (such as "iamauthorization" and "gateway") have higher usage. "Looking at CPU quota usage, the 'iamauthorization' service appears to be very close to its CPU limit (99.82%), indicating that the service has reached its resource limits and may be causing performance issues."

According to the analysis we made here, the answer we expect is that the "iamauthorization" service has reached the CPU upper limit and therefore the response times of the service are high.

Fig. 4, Fig. Based on the 5 images, the result obtained by the AI is as follows.

"There are graphs of average response times of two different services. In both graphs, increases in average response times were observed towards the end of the test (between 15:42 and 15:44). This may indicate that the performance of these services decreases under load. It has been stated that the maximum response times exceed 7 seconds, especially at the `/iamusers/api/v1/users` and `/iamauthorization/api/v1/self-resources/menus` endpoints. "This indicates that services are operating at less than optimal performance and perhaps some improvements or resource increases are required."

In our analysis here, the moments when the response times of the services suddenly increased were identified and reported. The analyzes made with AI directly match the inferences we make.

Since the creation of the jmx file depends on more than one metric, preparing the correct question set and reaching the final result requires determining the question combination. The jmx file created varies depending on the answers given to the question set we have prepared here. Since the jmx file given by AI is in xml format, it shows that arrangements must be made to open the file correctly in jmeter.

```
<TestPlan guiclass="TestPlanGui" testclass="TestPlan"
        <stringProp name="ThreadGroup.num_threads">30</stringProp>
        <stringProp name="ThreadGroup.ramp_time">1</stringProp>
        <boolProp name="ThreadGroup.scheduler">true</boolProp>
        <stringProp name="ThreadGroup.duration">300</stringProp>
        <stringProp name="ThreadGroup.delay"></stringProp>
```

In the answers given here, while the file was expected to arrive as xml, the "TestPlan" part of the script was directly output in line with the answers given. In Jmeter, inter-service integrations are costly because they are done manually.

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.5">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">
      <stringProp name="TestPlan.comments"></stringProp>
```

```
<boolProp name="TestPlan.functional_mode">false</boolProp>
<boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
<boolProp name="TestPlan.serialize_threadgroups">true</boolProp>
```

While the expected output should be as in the code block above, Thread properties information is provided separately by the AI. Following these experiments, it is expected that more accurate results will be obtained by improving the question sets in the next study.

ACKNOWLEDGMENT

I would like to thank Ethem Utku AKTAŞ for supporting me in every way in this study.

REFERENCES

[1] A. Avritzer and E.J. Weyuker, "Deriving Workloads for Performance Testing", *SoftwarePractice and Experience*, vol. 26, no. 6, pp. 613-633, June 1996.

[2] F.I. Vokolos and E.J. Weyuker, "Performance Testing of Software Systems", *Proc. ACM Workshop Software and Performance (WOSP 98)*, pp. 80-87, 1998-Oct.

[3] Junjie Chen, Guancheng Wang, Dan Hao, Yingfei Xiong, Hongyu Zhang, and Lu Zhang. 2019. History-guided configuration diversification for compiler testprogram generation. In 2019 34th IEEE/ACM International Conference on Auto-mated Software Engineering (ASE). IEEE, 305–316

[4] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Communications of the ACM 53(4), 50–58 (2010).