Enhance Software Performance Testing with Artificial Intelligence

Rini Susan V S

rinisvs@gmail.com

Abstract

With advances in data collection, processing, and computation, AI has become the latest buzzword in every industry. Performance testing is a type of testing in which the speed, responsiveness, and stability of a software, product, or network is evaluated under peak workload. Performance testing has evolved a lot over time. The focus has shifted from mere testing to performance engineering aspects in which the testing teams identify bottlenecks, perform error analysis, and provide performance tuning recommendations.

To keep up with the agile mode of development, the traditional testing process is no longer adequate, and teams need to bring in automation. Artificial Intelligence can play an important factor in test automation to reduce the time consumption and manual intervention involved in various test phases. Generative Artificial Intelligence, a subset of AI can aid in these activities. This paper discusses how performance testing and engineering can benefit from Artificial Intelligence and provides some use cases.

Biography

Rini Susan V S is a senior quality engineer focusing on Software Performance Testing and Engineering. She has an extensive working knowledge of Performance testing, Application monitoring, DevOps, Profiling, and CI/CD tools. Rini is a regular contributor to technical blogs and articles and has co-authored an article published on developer.com. She has completed a technical program in Artificial Intelligence and Machine Learning.

1 Introduction

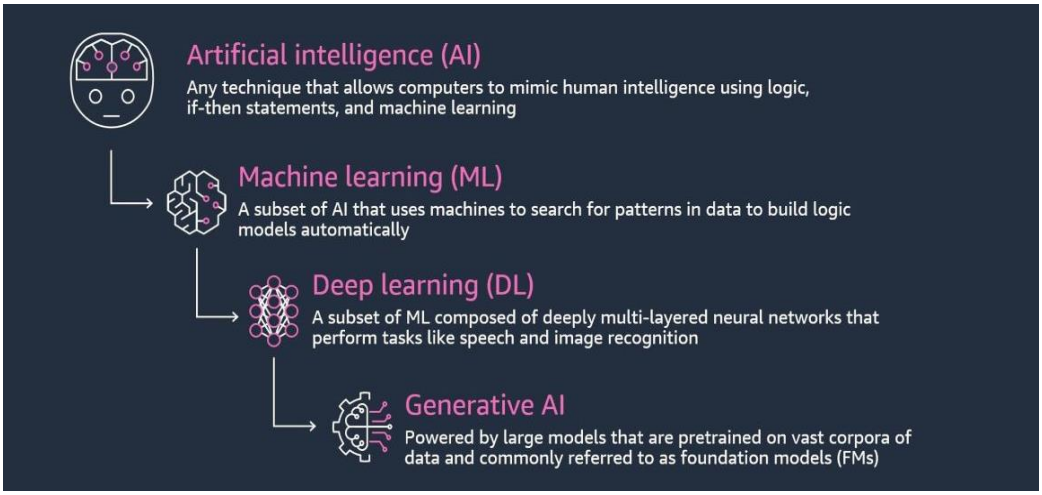

Fig. 1: Relationship between AI, ML, DL, and Gen AI.

Artificial intelligence or AI, is the technology that enables computers and machines to simulate human intelligence and problem-solving capabilities[1]. Today, there are many, real-world applications of AI systems including, speech recognition, online chatbots, computer vision, robotics, etc. Machine learning or ML, is a field of study in Artificial Intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions[2]. ML can be used to reduce the amount of routine and tedious tasks in software development and testing. Generative Artificial Intelligence (GenAI) is a subset of Machine Learning. Generative AI models use neural networks to identify the patterns and structures within existing data to generate new and original content[3].

Software Testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate[4]. Performance testing is a type of testing intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload[5]. Performance testing checks and validates an application's capacity and ensures that it works well within the acceptable Service Level Agreements.

Digital customers are provided with a lot of choices these days and performance testing plays a major role in ensuring customer satisfaction. If the performance of an application doesn’t meet the expectations of customers, they will opt for another application from a competitor, resulting in business loss for companies.

2 Software Performance Testing Progression

Performance testing has evolved over time and the focus has shifted to performance engineering. Earlier, performance testing teams were mainly expected to conduct load test executions and share the test results with development teams. These days, performance engineering aspects are also sought after and the testing teams are expected to identify bottlenecks, perform error analysis, and provide performance-tuning recommendations.

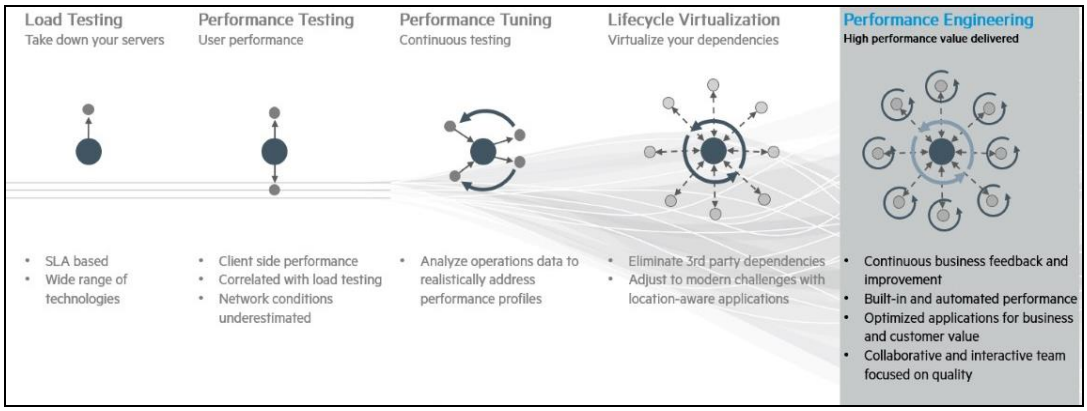

Fig. 2: Evolution of Performance Engineering

As enterprise software platforms become more complex, performance issues have become a serious risk that results in loss of millions of dollars. To keep up with the agile mode of development, the traditional testing process is no longer adequate, and teams need to bring in automation. Artificial intelligence can play an important role in test automation by reducing time consumption and manual intervention in various test phases. Machine Learning and Generative AI can aid in these activities.

3 Machine Learning in Performance Testing and Performance Engineering

Machine learning solutions can evaluate and interpret thousands of statistics per second, providing real-time insight into a system's behavior. Machine learning algorithms can identify data patterns, build statistical models, and make predictions. ML-based anomaly detection systems can help to identify performance bottlenecks faster and more accurately.

Generative AI is a subset of Machine Learning that uses AI to create new content in text, image, audio, and video format. It is powered by Foundation Models that can multi-task and perform tasks including summarization, classification, image generation, and live chat. These models can be tailored for targeted use cases with minimal training.

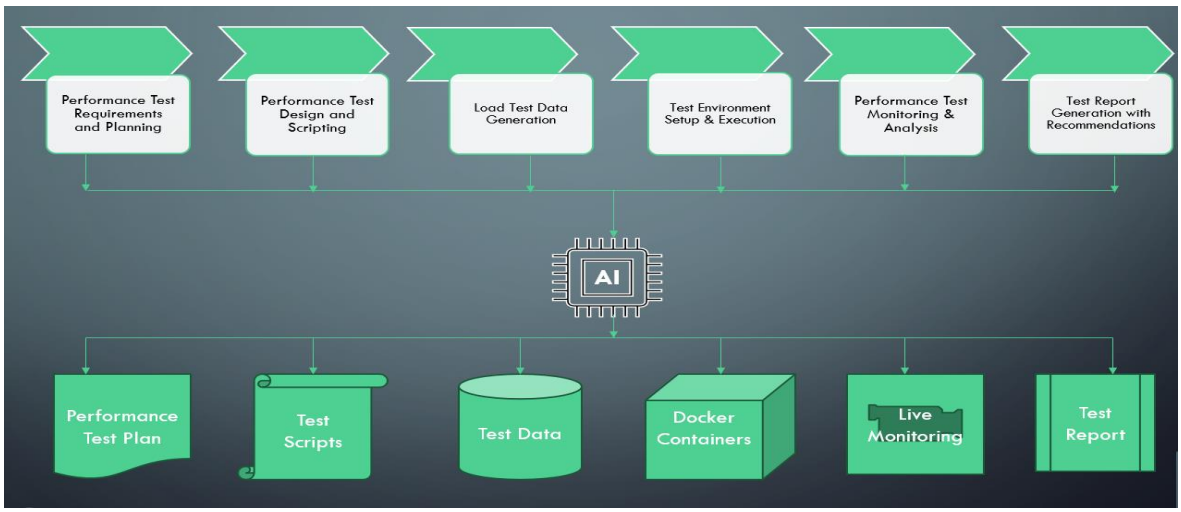

Fig. 3: AI Use Cases in Software Performance Testing Life Cycle

3.1 Performance Testing Use Cases

Some of the performance testing use cases that can be addressed using Generative AI models include

- Automatic generation of performance test plan, by providing basic information.
- Test data set creation in large volumes for load testing and endurance testing.
- Automatic test script generation for load testing tools like Gatling, by providing the required information.
- Test report generation in the desired format type, automatically after test execution.

3.2 Performance Engineering Use Cases

Some of the performance engineering use cases that can be addressed using Machine Learning techniques include

- Predict server utilization based on metrics like CPU usage, memory utilization, and amount of disk reads and writes.
- Detect outliers in transaction response time based on transaction logs over a period of time.
- Forecast load during special events like holiday or anniversary sales, with historical data as input from various data sources.

4 Potential AI Testing Solutions

4.1 Machine Learning Models

Large Language Models (LLM) and Machine Learning algorithms (like mentioned below), can be used to automate testing activities, identify performance bottlenecks, and improve the overall quality of software performance testing.

- Foundational Models like Amazon Titan, Anthropic's Claude, Google Gemini, or OpenAI GPT models for test plan and test report generation.
- Mistral AI or Google PaLM 2 models for code generation for Gatling scripts in Scala.
- OpenAI DALL-E or Stable Diffusion XL models for image generation in test report creation.
- Artificial Neural Network (ANN) models to predict server utilization based on metrics like CPU usage, memory utilization, and amount of disk reads and writes.
- K-nearest neighbor (KNN) algorithm to detect anomalies in transaction response time and server utilization matrices. [6].
- State Space Model for forecasting during special events or holiday/anniversary sales, with historical data as input from various data sources.

4.2 AI-Based Testing Tools

Various licensed and open-source toolkits with machine-learning capabilities can be integrated with existing performance testing tools to optimize the testing process. There are also advancements in AI-driven automation testing tools. Some emerging testing tools with AI capabilities are mentioned below.

- Applitools uses Visual AI and no-code approaches for automated UI testing [7].
- Mabl tool uses a low code approach and supports automation, accessibility, and performance testing. [8].
- Functionize tool supports UI, API, and Database testing and provides a cloud testing platform [9].

5 Solution Implementation

5.1 Use Case: Performance Test Report Generation

Gemini-1.0-pro, a Google AI model, is utilized here to create performance test reports. The main idea behind this use case is to reduce the overall testing effort by automating the manual report creation activity.

```
# Create model with the selected model name and configs
model = genai.GenerativeModel(model_name='gemini-1.0-pro',
                              generation_config=generation_config,
                              safety_settings=safety_settings)
```

Fig. 4: Model Creation

Performance test result files are given as input to the model in CSV format. The test report is created based on the prompt provided to the model, according to the project requirement.

```
# Read input file
url = "Test_Results.csv"
with open(url) as file:
  input_file = file.read()

# Print first 5 lines
for line in input_file.splitlines()[:5]:
  print(line)
```

```
timeStamp,elapsed,label,responseCode,responseMessage,threadName,dataType,success,failureMessage,bytes,sentBytes,grpThreads,allTh
reads,URL,Latency,IdleTime,Connect
1652128008147,16,Test_01_Search_FirstName,200,HTTP/1.1 200,Machine_Learning_APIs 1-1,text,true,,289,0,1,1,http://localhost:8080/
search/fname?firstName=Mary,15,0,0
1652128008240,9,Test_03_Search_Email,200,HTTP/1.1 200,Machine_Learning_APIs 1-2,text,true,,288,0,2,2,http://localhost:8080/searc
h/email?email=alexwill%40gmail.com,8,0,0
1652128008348,4,Test_05_Search_AccountId,200,HTTP/1.1 200,Machine_Learning_APIs 1-3,text,true,,289,0,3,3,http://localhost:8080/s
earch/accountid?accountId=1634589017,4,0,0
1652128008458,3,Test_01_Search_FirstName,200,HTTP/1.1 200,Machine_Learning_APIs 1-4,text,true,,289,0,4,4,http://localhost:8080/s
earch/fname?firstName=Mary,3,0,0
```

```
# Generate content for the prompt and display the execution time
%%time
response5 = model.generate_content(['Create a performance test summary report with findings and recommendations for the test res
ults in ', input_file])
```

```
CPU times: user 60.3 ms, sys: 2.08 ms, total: 62.4 ms
Wall time: 11.1 s
```

Fig. 5: Reading the test results file and prompt as input

Test summary report is generated in the required format with details of test results including response time and resource utilization metrics, test findings, and possible recommendations for further improvement. This solution is valuable for projects operating under stringent deadlines and budget constraints.

```
# Print the response - test summary report document
print(response5.text)
```

```
**Performance Test Summary Report**

**Test Results:**

* **Total Tests:** 100
* **Successful Tests:** 100
* **Failed Tests:** 0

**Response Time Metrics:**

* **Average Response Time:** 15 ms
* **Median Response Time:** 14 ms
* **90th Percentile Response Time:** 18 ms
* **95th Percentile Response Time:** 20 ms
* **99th Percentile Response Time:** 25 ms

**Resource Utilization Metrics:**

* **Average CPU Utilization:** 5%
* **Average Memory Utilization:** 10%

**Findings:**

* The API performed consistently well under load, with all requests completing successfully.
* Response times were within acceptable limits, with the majority of requests completing in under 20 ms.
* Resource utilization was low, indicating that the API can handle a higher load without performance degradation.

**Recommendations:**

* Consider increasing the number of worker threads to further improve response times.
* Monitor resource utilization closely to ensure that the API remains performant under peak load.
* Implement caching mechanisms to reduce the load on the database.
```

Fig. 6: Performance Test Summary Report

6 Road Ahead

6.1 Possible Directions

In the near future, we can anticipate a surge in AI-powered tools offering advanced testing features. AI can enhance these tools by incorporating more performance testing and performance engineering capabilities. Following are some key performance engineering use cases.

- Detect anomalies in transaction response time or server utilization
- Root cause analysis of performance issues
- Forecast application behavior based on previous events or logs

6.2 Factors to Consider

Though Artificial intelligence (AI) has been widely accepted across industries, implementing AI is not a straightforward task. Organizations must examine various aspects to come up with a well-defined AI strategy. The following are some key factors to consider while implementing AI solutions.

- Hallucinations: AI models are not always perfect and may generate inaccurate/false information, content, or facts. This is known as hallucination and can lead to mistakes and risks.
- Integration with systems: Integration of Artificial Intelligence capabilities into existing systems requires significant planning. The feasibility of AI solutions for the project-specific requirements must be evaluated, along with its impact on existing systems.
- Model choice: Organizations can create AI models or use any of the available AI models. The model is only as good as the data it is trained on; hence it is critical to have correct and pertinent data for model training.
- Computational resources: While selecting an AI model, the usage of computational resources required is also vital. Based on the AI workload, and the model choice, sometimes high-performing Graphics Processing Units (GPUs) may be required.

6.3 Responsible AI

While the potential of AI is immense, these technologies can also raise critical challenges that need to be addressed thoughtfully, and carefully. Responsible development is essential to mitigate the unintended or unforeseen consequences due to the misuse of AI. The following are some key Responsible AI objectives that must be prioritized. [10].

- Be socially beneficial: The AI solution or tool is intended to benefit software testing professionals in terms of time and effort, thereby reducing the overall project budget.
- Be built and tested for safety: Safety is a major concern of Generative AI applications using prompt engineering. Hence safety must be ensured by setting the blocking levels of harmful contents based on project requirements.
- Be accountable to people: The solutions or tools are intended to aid software performance testers. The feedback from testers must be sought and incorporated for further tool improvement.

7 Conclusion

Given the transformative power of AI, it is crucial to keep up with its advancements in this fast-paced technological era. Artificial Intelligence technologies are essential in the field of software performance testing as well; incorporating them can help to enhance testing efficiency and accuracy.

Artificial Intelligence techniques can transform software performance testing from manual, time-consuming activities to automated, data-driven, predictive insights. Generative AI solutions can help to automate labor-intensive tasks involved in the performance testing lifecycle, while Machine Learning capabilities can aid in performance forecasting and anomaly detection.

To sum up, individuals or teams with foresight can utilize Artificial Intelligence and Machine Learning technologies to enhance software performance testing processes and prevent performance issues, rather than reacting after they occur. Consequently, organizations can shift from a reactive to a proactive performance testing strategy.

References

[1] IBM. “What is artificial intelligence (AI)?”, http://www.ibm.com/topics/artificial-intelligence.html

[2] Wikipedia. “Machine Learning”, http://en.wikipedia.org/wiki/Machine_learning.html

[3] NVIDIA. “Generative AI”, http://www.nvidia.com/en-us/glossary/generative-ai.html

[4] B. Meyer. 2018. Seven Principles of Software Testing

[5] Microsoft Docs. 2010. “Chapter 1 – Fundamentals of Web Application Performance Testing.”, Performance Testing Guidance for Web Applications

[6] IBM. “Anomaly detection in machine learning: Finding outliers for optimization of business functions”, http://www.ibm.com/blog/anomaly-detection-machine-learning.html (accessed December 19, 2023).

[7] Applitools. https://applitools.com/

[8] Mabl. https://www.mabl.com/

[9] Functionize. https://www.functionize.com/

[10] Google AI. “Responsibility”, https://ai.google/responsibility/principles/

[Fig1] Amazon Web Services: http://aws.amazon.com/getting-started/decision-guides/machine-learning-on-aws-how-to-choose/.html

[Fig2] Todd DeCapua, Shane Evans: Effective Performance Engineering, ISBN: 9781491950869, 2016