Kalinga University

Faculty of Computer Science & IT

Course- BCAAIML
Subject- Software Engineering and Testing
Course Code – BCAAIML503 **Sem- 5th**

Unit V

Performance Testing

Performance testing is a type of software testing that focuses on evaluating the performance and scalability of a system or application. Performance testing aims to identify bottlenecks, measure system performance under various loads and conditions, and ensure that the system can handle the expected number of users or transactions.

Types of performance testing

Performance Testing is the process of analyzing the quality and capability of a product. It is a testing method to determine the system's performance in terms of speed, reliability, and stability under varying workloads.
Just as performance testing is essential for ensuring that software applications can handle expected workloads and maintain stability under various conditions, understanding the intricacies of this testing type is crucial for delivering high-quality software.

1. Load testing
Load testing simulates a real-world load on the system to see how it performs under stress. It helps identify bottlenecks and determine the maximum number of users or transactions the system can handle. It checks the product's ability to perform under anticipated user loads. The objective is to identify performance congestion before the software product is launched in the market.
2. Stress testing
Stress testing is a type of load testing that tests the system's ability to handle a high load above normal usage levels. It helps identify the breaking point of the system and any potential issues that may occur under heavy load conditions. It involves testing a product under extreme workloads to see whether it handles high traffic or not. The objective is to identify the breaking point of a software product.
3. Spike testing
Spike testing is a type of load testing that tests the system's ability to handle sudden spikes in traffic. It helps identify any issues that may occur when the system is suddenly hit with a high number of requests. It tests the product's reaction to sudden large spikes in the load generated by users.

4. Soak testing

Soak testing is a type of load testing that tests the system's ability to handle a sustained load over a prolonged period. It helps identify any issues that may occur after prolonged usage of the system.

5. Endurance testing

Endurance testing is similar to soak testing, but it focuses on the long-term behavior of the system under a constant load. It is performed to ensure the software can handle the expected load over a long period.

6. Volume testing

In Volume testing , a large number of data is saved in a database and the overall software system's behavior is observed. The objective is to check the product's performance under varying database volumes.

7. Scalability testing

In Scalability testing , the software application's effectiveness is determined by scaling up to support an increase in user load. It helps in planning capacity additions to your software system.

Why use performance testing?

- The objective of performance testing is to eliminate performance congestion.
- It uncovers what needs to be improved before the product is launched in the market.
- The objective of performance testing is to make software rapid.
- The objective of performance testing is to make software stable and reliable.
- The objective of performance testing is to evaluate the performance and scalability of a system or application under various loads and conditions.
- It helps identify bottlenecks, measure system performance, and ensure that the system can handle the expected number of users or transactions.
- It also helps to ensure that the system is reliable, stable, and can handle the expected load in a production environment.

Advantages of Performance Testing

- **Identifying bottlenecks** : Performance testing helps identify bottlenecks in the system such as slow database queries, insufficient memory, or network congestion. This helps developers optimize the system and ensure that it can handle the expected number of users or transactions.
- **Improved scalability:** By identifying the system's maximum capacity, performance testing helps ensure that the system can handle an increasing number of users or transactions over time. This is particularly important for web-based systems and applications that are expected to handle a high volume of traffic.
- **Improved reliability:** Performance testing helps identify any potential issues that may occur under heavy load conditions, such as increased error rates or slow response times. This helps ensure that the system is reliable and stable when it is deployed to production.
- **Reduced risk:** By identifying potential issues before deployment, performance testing helps reduce the risk of system failure or poor performance in production.

- **Cost-effective:** Performance testing is more cost-effective than fixing problems that occur in production. It is much cheaper to identify and fix issues during the testing phase than after deployment.
- **Improved user experience** : By identifying and addressing bottlenecks, performance testing helps ensure that users have a positive experience when using the system. This can help improve customer satisfaction and loyalty.
- **Better Preparation:** Performance testing can also help organizations prepare for unexpected traffic patterns or changes in usage that might occur in the future.
- **Compliance** : Performance testing can help organizations meet regulatory and industry standards.
- Better understanding of the system: Performance testing provides a better understanding of how the system behaves under different conditions, which can help in identifying potential issue areas and improving the overall design of the system.
- Performance testing ensures the speed, load capability, accuracy, and other performances of the system.
- It identifies, monitors, and resolves the issues if anything occurs.
- It ensures the great optimization of the software and also allows many users to use it at the same time.
- It ensures the client as well as the end-customer's satisfaction. Performance testing has several advantages that make it an important aspect of software testing:

Disadvantages of Performance Testing

Performance testing also has some disadvantages, which include:

- **Resource-intensive:** Performance testing can be resource-intensive, requiring significant hardware and software resources to simulate many users or transactions. This can make performance testing expensive and time-consuming.
- **Complexity:** Performance testing can be complex, requiring specialized knowledge and expertise to set up and execute effectively. This can make it difficult for teams with limited resources or experience to perform performance testing.
- **Limited testing scope:** Performance testing is focused on the performance of the system under stress, and it may not be able to identify all types of issues or bugs. It's important to combine performance testing with other types of testing such as functional testing, regression testing, and acceptance testing.
- **Inaccurate results:** If the performance testing environment is not representative of the production environment or the performance test scenarios do not accurately simulate real-world usage, the results of the test may not be accurate.
- **Difficulty in simulating real-world usage:** It's difficult to simulate real-world usage, and it's hard to predict how users will interact with the system. This makes it difficult to know if the system will handle the expected load.
- **Complexity in analyzing the results:** Performance testing generates a large amount of data, and it can be difficult to analyze the results and determine the root cause of performance issues.

- Sometimes, users may find performance issues in the real-time environment.
- Team members who are writing test scripts or test cases in the automation tool should have a high level of knowledge.
- Team members should have high proficiency in debugging the test cases or test scripts.
- Low performances in the real environment may lead to the loss of a large number of users

Factors controlling the Performance Testing

Performance Testing is conducted to discover the response time, throughput, etc., and also to execute its required functions by doing a comparison with different versions of the same product or different competitive products. It is a type of software testing that ensures that software applications perform correctly under their expected workload. It is a testing technique carried out to determine system performance in terms of sensitivity, reactivity, and stability under a particular workload.

The factors controlling the performance testing are given below :

1. **Throughput –**

The throughput is a concept of determining how well a software or an application can perform. It is **to handle** the capability of the system or the product handling multiple transactions that are determined by a factor. It also represents the number of requests or business transactions processed by the software or an application in the specified time frame. Note that the throughput, i.e., the number of transactions serviced by the product per unit time varies according to the load the product is put under.

2. **Response Time –**

The response time is defined as the delay between the point of request and the first response from the software product. In a normal client-server environment, a throughput determines the number of transactions that can be handled by the server whereas a response time defines the delay between the request and response of an application.

3. **Latency –**

The latency can be defined as the delay caused by an application, OS, and environment that is calculated separately. Note that, not all the delay that happens between the request and the response is caused due to the product. In the networking scenario, the network or other products that are sharing the network resources can cause delays. Therefore, we must know that what delays the application causes and what delays the environment causes.

4. **Tuning –**

Tuning is a process in which the product's performance is improved by adding some different values to the parameters (variables) of the application, operating system, and other components.

It enhances the product's performance without modifying the source code of the product and here each product may have certain parameters that can be added at run time to get a great performance.

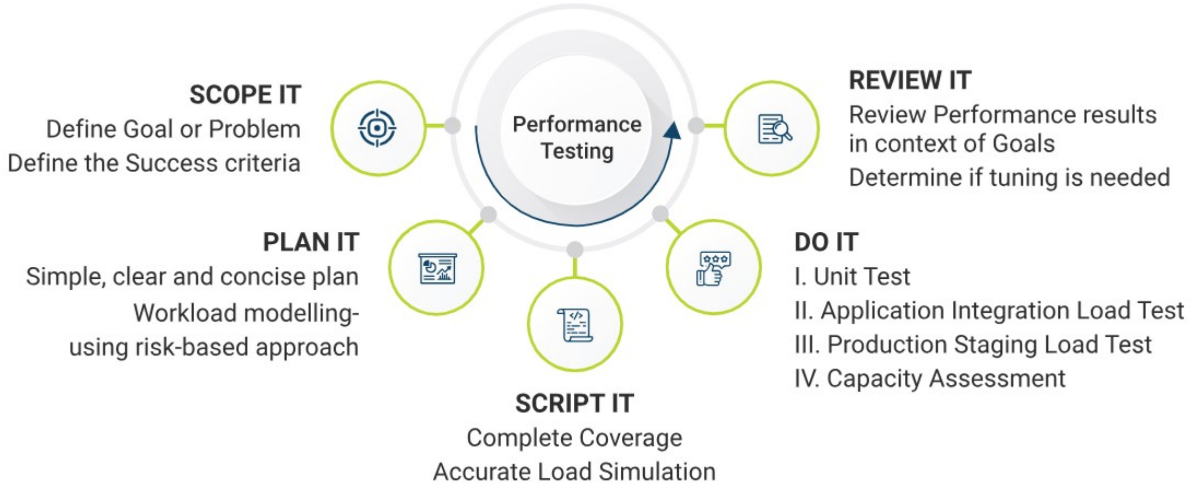
5. **Benchmarking –**

Benchmarking can be defined as the process of comparing the throughput and response time of the product to those of competitive products. Here, no two products are the same in features, cost, and functionality. Therefore, it is not easy to decide what two or more parameters must be compared between the two products.

6. **Capacity Planning –**

Capacity planning is a process in which we determine what type of hardware and software resources are needed for running an application with a given user load. Here, the most significant factor that affects performance testing is the availability of resources. The right kind of hardware and software configuration is needed to derive the best results from performance testing and deployments.

Performance Testing Methodology


Performance Testing
SCOPE IT
Define Goal or Problem
Define the Success criteria
REVIEW IT
Review Performance results in context of Goals
Determine if tuning is needed
PLAN IT
Simple, clear and concise plan
Workload modelling- using risk-based approach
DO IT
I. Unit Test
II. Application Integration Load Test
III. Production Staging Load Test
IV. Capacity Assessment
SCRIPT IT
Complete Coverage
Accurate Load Simulation

One of the first steps in this planning is to build an accurate workload model. This model will be representative of user behavior and user load. If the workload model is incorrect, then the results of the performance test could be misleading, so it's very important for those planning the tests to understand the current or expected usage and utilization of the system and its resources. For instance, some functions, such as administration ones, are rarely used and don't generally need to be included in the performances tests. QA Mentor uses a risk-based approach to select what should be tested and what can be left out.

Everything is laid out in simple terms in the test plans, including but not limited to:

– Connections that will be used
– Number of servers and their specifications
– Number of requests, users, or transactions
– Testing tools
– Test scenarios
– Ramp up period
– File sizes
– Think time
– Failure criteria
– Success criteria
– Reporting tools

Script It

The testers writing the scripts buckle-down and learn the system thoroughly at this stage. They make sure the system can be tested using the chosen tool, while learning the business processes. Regardless of the tool chosen to script and execute the tests, the most crucial factor is the simulation. Testers spend time to make sure the scripts provide complete coverage, simulate the load accurately, and that they run correctly after all coding changes. After spending time in the system, our performance testers will be able to identify possible problem areas due to noting response times for each step of the scripts. Test data is created or identified at this phase as well.

Do It

We've planned the execution, next we execute the plan. On a system being newly built, the performance testing is phased in conjunction with the development.

Phase I – Unit Test – Generally this phase of performance testing is completed by developers after they complete components and before integrating them. Performance testing at this phases uses tools such as a memory profiler, code profiler, and coverage profiler. Each of these tools can help identify an area at risk for significant issues when many users are accessing it at the same time. Testing and fixing at this stage is much easier and less expensive.

Phase II – Application Integration Load Test – After the functional integration test has been completed and any functional issues are fixed, the integrated load test can begin. This phase is a full load test using the planned number of users to simulate an expected production load. For best results, the test is performed twice. The first time through, the test is executed with minimal monitoring. At this point, we just want to make sure it can hold up to an expected load. The second time through we add detailed monitoring. At this point we should be able to identify any bottlenecks and begin performance tuning.

Phase III – Production Staging Load Test – At this point, the team has moved the application to an environment that mimics production and performed functional testing. This phase of load testing is where we really see how the application handles in a shared environment where resources are in competition. We turn up the heat in this phase and make sure the system can handle the expected production traffic.

Phase IV – Capacity Assessment – Once we've come to this point, we know that the application or system meets expectations, but we want to go one step further. This phase determines the system's breaking point. By assessing the capacity early, you can set points in the future when you know new hardware will need to be added in the future, or plan for new architecture. During this test we steadily increase the load on the system until resources are saturated, response times slow, and data transfer rates drop. The information obtained during this test will aid in capacity planning for the future of the application.

Performance Testing Tools

1. Apache JMeter: is an open-source tool used for performance testing and load testing of applications. It simulates multiple users sending requests to a server, collecting performance metrics to analyze the application's behavior under different load conditions.
2. Open STA (Open, Systems Testing Architecture) Apache JMeter is an open-source tool used for performance testing and measuring the load and stress on web applications. Its main function is to simulate multiple users accessing a site simultaneously to evaluate its performance and scalability.
3. Load Runner: LoadRunner is a performance testing tool used to simulate virtual users and analyze the behavior of applications under load. It helps identify performance bottlenecks by measuring system performance and response times under varying conditions.
4. Web Load: Web Load is a performance testing tool designed to test the scalability and reliability of web applications under various load conditions. The function by sending user requests to a web server, which processes the requests and sends back the appropriate web pages or data.
5. Gatling: Gatling is an open-source load testing tool designed to analyze and measure the performance of web applications. Its main function is to simulate a large number of users interacting with a website to identify performance bottlenecks and ensure the site can handle high traffic.
6. BlazeMeter: BlazeMeter is a cloud-based testing platform designed to simulate large-scale user load and measure application performance. It allows developers and testers to run continuous performance tests and analyze results to ensure their applications can handle high traffic and perform optimally under stress.

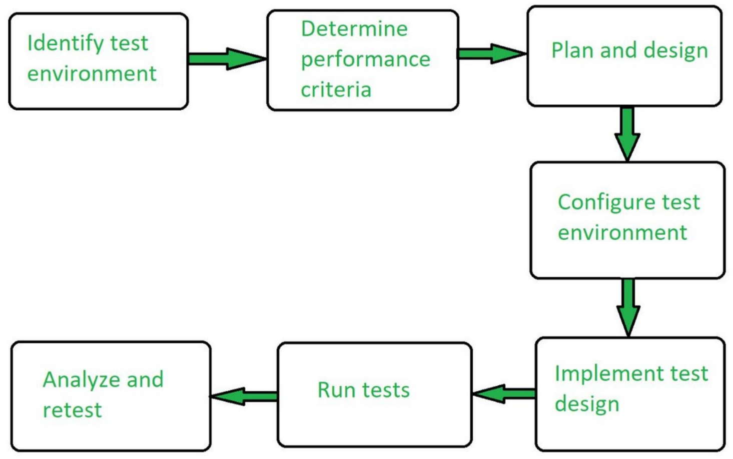
Performance Testing Attributes

- **Speed:** It determines whether the software product responds rapidly.
- **Scalability:** It determines the amount of load the software product can handle at a time.

- **Stability:** It determines whether the software product is stable in case of varying workloads.
- **Reliability:** It determines whether the software product is secure or not.

Process of performance testing

Conducting performance testing involves several steps to ensure that a software application can handle expected loads and perform well under stress. Here's a simplified guide on how to conduct performance testing:


Identify test environment
Determine performance criteria
Plan and design
Configure test environment
Implement test design
Run tests
Analyze and retest

Step 1: Set Up the Testing Environment First prepare the place where you will run the tests for the performance testing. Make sure you have all the needed tools and understanding for the setup, like what devices and software you will be using for the performance testing.

Step 2: Decide What to Measure Think about what you want to know from the tests. This will include things like how fast the system responds to and how much it can handle the tests. You can also look at successful similar systems to set your goals.

Step 3: Plan Your Tests to Figure out different scenarios to test, considering things like how users might behave and what data you will use. This helps you create tests that cover a range of situations and decide what data to collect.

Step 4: Set Up Your Tools Get everything ready for testing, including tools and ways to track what's happening during the tests.

Step 5: Create and Run Tests Make the tests based on your plan and run them. Keep track of all the data you get from the tests.

Step 6: Look at the Results After each test, see what you find out. Adjust your tests based on what you learn, and run them again to see if things change.

Step 7: Keep Testing Keep analyzing and adjusting your tests to get the best results. Repeat the process until you are satisfied with the performance.

Regression Testing

Regression testing is like a **software quality** checkup after any changes are made. It involves running tests to make sure that everything still works as it should, even after updates or tweaks to the code. This ensures that the software remains reliable and functions properly, maintaining its integrity throughout its development lifecycle.
What is Regression Testing

When to do regression testing

- When new functionality is added to the system and the code has been modified to absorb and integrate that functionality with the existing code.
- When some defect has been identified in the software and the code is debugged to fix it.
- When the code is modified to optimize its working.

Types of Regression Testing

There are 4 types of regression testing in software testing. Let's discuss them one by one:

1. Corrective Regression testing –

This type of testing is used in the software testing process when specifications are not modified and all test cases can be reused in the testing. It requires less time to find the faults or bugs. It is known as one of the most popular types in the current generation and also for its convenience and repetitive use in testing.
This testing technique analyses the effect of new code on the product's existing source code and also uses a subset of existing test cases to minimize the costs and efforts needed for testing.

2. Progressive Regression testing –

This type of testing is used in the software testing process when the specifications are modified or changed and all the new test cases must be created newly. This testing is recommended when you are developing new test cases.
It also allows the testers to perform the required step within the modified or updated version of the program without using the current program code. The testing also works fine when there are only a few changes to be performed in the model and also while creating new test cases.

3. Retest-all Regression testing –

This type of testing is used in the software testing process and it reuses all tests but this method of testing may require more time and cost as it does the unnecessary execution of tests. Therefore, it is advised for the testers to know and understand the activity before starting the testing process.

However, when the modification to a system is small, this method of testing is not recommended. Also, It is not recommended to perform this testing for every software product because of time constraints as most of the clients prefer avoiding this.

4. Selective Regression testing –
This type of testing is used in the software testing process that uses a subset of the existing test cases to reduce the time and cost of the testing. The objective of this testing is to find the dependencies between a test case and the program entities it covers. Selective regression uses the following steps in its testing process:

- **Step 1**. The first step is to identify the affected components of the software after the program has been changed.
- **Step 2**. After that, it selects a subset of test cases from an existing test suite(set of test cases) that covers the components of software affected by the changes or modifications.
- **Step 3.** Then, it tests the changed program to create the correctness of the program.
- **Step 4.** After that, it examines the test results to identify the software failure.
- **Step 5.** If it finds any fault then it corrects that fault(s) that causes that failure.
- **Step 6**. At last, it updates the test suite and test history of the program.

Advantages of Regression Testing

- Automated unit testing
- Comprehensive test coverage
- System integration
- Faster test execution completion
- Improved developer productivity
- Parallel testing
- Reduced costs
- Regression testing improves product quality
- Reusability
- Scalability
- Time efficiency

Disadvantages of Regression Testing

- It can be time and resource-consuming if automated tools are not used.
- It is required even after very small changes in the code.
- Time and Resource Constraints.
- Among the significant risks associated with regression testing are the time and resources required to perform it.

- Incomplete or Insufficient Test Coverage.
- False Positives and False Negatives.
- Test Data Management Challenges.