LOAD TESTING

Research on Load Testing Tools

Submitted By:

Nischal Shrestha

Submitted To:

Prajesh Shakya

Date: 10th April, 2025

ABSTRACT

This report summarizes the critical practice of load testing in evaluating the performance and stability of software applications and infrastructure under anticipated and peak user loads. As digital services become increasingly integral, ensuring their resilience and responsiveness is really important for maintaining user satisfaction and business continuity. This report details various load testing methodologies and tools employed to simulate realistic user traffic and identify potential bottlenecks, performance degradation, and failure points within a system. It explores different types of load tests, including stress testing, soak testing, and spike testing, and their respective objectives in assessing system behavior under varying conditions.

TABLE OF CONTENTS

LOAD TESTING

Introduction

Load Testing is a type of Performance Testing that determines the performance of a system, software product, or software application under real-life-based load conditions. It determines the behavior of the application when multiple users use it at the same time. It is the response of the system measured under varying load conditions.The load testing is carried out for normal and extreme load conditions.

- Load testing is a type of performance testing that simulates a real-world load on a system or application to see how it performs under stress.
- The goal of load testing is to identify bottlenecks and determine the maximum number of users or transactions the system can handle.
- It is an important aspect of software testing as it helps ensure that the system can handle the expected usage levels and identify any potential issues before the system is deployed to production.

During load testing, various scenarios are simulated to test the system's behavior under different load conditions. This can include simulating a high number of concurrent users, simulating numerous requests, and simulating heavy network traffic. The system's performance is then measured and analyzed to identify any bottlenecks or issues that may occur.

Load Testing Techniques

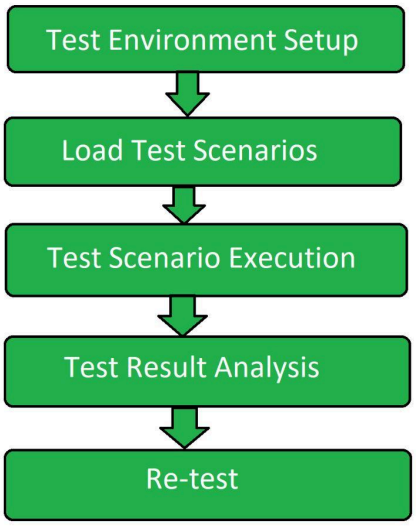
1. Stress testing: Testing the system's ability to handle a high load above normal usage levels
2. Spike testing: Testing the system's ability to handle sudden spikes in traffic
3. Soak testing: Testing the system's ability to handle a sustained load over a prolonged period of time
4. Tools for Performance Testing: Make use of specialized load testing tools like Locust, Gatling, JMeter, LoadRunner, and Apache Benchmark. These tools assist in gathering performance measurements and simulating a large number of users.

5. Specify the Test Objectives: Clearly state what your load test's goals are. Recognize the required response times, transaction volumes and expected user behavior.
6. Determine Crucial Situations: Determine the essential user scenarios that correspond to common usage patterns. A variety of actions, including user logins, searches, form submissions and other significant interactions, should be covered by these scenarios.

Objectives of Load Testing

1. Evaluation of Scalability: Assess the system's ability to handle growing user and transaction demands. Find the point at which the system begins to function badly.
2. Planning for Capacity: Describe the system's ability to accommodate anticipated future increases in the number of users, transactions and volume of data. Making well-informed decisions regarding infrastructure upgrades is made easier by this.
3. Determine bottlenecks: Identify and localize bottlenecks in the application or infrastructure's performance. Finding the places where the system's performance can suffer under load is part of this.
4. Analysis of Response Time: For crucial transactions and user interactions, track and evaluate response times. Make that the system responds to changes in load with reasonable response times.
5. Finding Memory Leaks: Find and fix memory leaks that may eventually cause a decline in performance. Make sure the programme doesn't use up too many resources when it's running.

Load Testing Process



1. Test Environment Setup: Firstly create a dedicated test environment setup for performing the load testing. It ensures that testing would be done in a proper way.
2. Load Test Scenario: In the second step load test scenarios are created. Then load testing transactions are determined for an application and data is prepared for each transaction.
3. Test Scenario Execution: Load test scenarios that were created in the previous step are now executed. Different measurements and metrics are gathered to collect the information.
4. Test Result Analysis: Results of the testing performed are analyzed and various recommendations are made.
5. Re-test: If the test is failed then the test is performed again in order to get the result in the correct way.

Metrics of Load Testing

Metrics are used in knowing the performance of load testing under different circumstances. It tells how accurately the load testing is working under different test cases. It is usually carried out after the preparation of load test scripts/cases. There are many metrics to evaluate the load testing. Some of them are listed below.

1. Average Response Time

It tells the average time taken to respond to the request generated by the clients or customers or users. It also shows the speed of the application depending upon the time taken to respond to all requests generated.

2. Error Rate

   The Error Rate is mentioned in terms of percentage denoting the number of errors occurred during the requests to the total number of requests. These errors are usually raised when the application is no longer handling the request at the given time or for some other technical problems. It makes the application less efficient when the error rate keeps on increasing.

3. Throughput

   This metric is used in knowing the range of bandwidth consumed during the load scripts or tests and it is also used in knowing the amount of data which is being used for checking the request that flows between the user server and application main server. It is measured in kilobytes per second.

4. Requests Per Second

   It tells how many requests are being generated to the application server per second. The requests could be anything like requesting images, documents, web pages, articles or any other resources.

5. Concurrent Users

   This metric is used to take the count of the users who are actively present at the particular time or at any time. It just keeps track of those who are visiting the application at any time without raising any request in the application. From this, we can easily know that at which time the high number of users are visiting the application or website.

6. Peak Response Time

   Peak Response Time measures the time taken to handle the request. It also helps in finding the duration of the peak time(longest time) at which the request and response cycle is handled and finding which resource is taking longer to respond to the request.

Load Testing Tools

1. Apache Jmeter: is an open-source tool used for performance testing and measuring the load and functional behavior of web applications. It simulates multiple users sending

requests to a web server, analyzes the server's response, and measures performance metrics such as response time, throughput, and resource utilization.

2. WebLoad: It is a performance testing tool designed to simulate user load on web applications and measure their behavior under various conditions. It helps identify performance bottlenecks and ensure that web applications can handle expected traffic.
3. NeoLoad: It is a performance testing tool used to simulate user traffic and measure how well applications handle load and stress. It helps identify bottlenecks and performance issues by generating virtual users to test the application's scalability and reliability.
4. LoadNinja: It is a cloud-based performance testing tool that enables users to simulate real-world user loads on their applications. It provides detailed insights into application performance and scalability by running load tests in real-time, helping teams identify and resolve performance bottlenecks.
5. HP Performance Tester: HP Performance Tester, now known as Micro Focus LoadRunner, is a performance testing tool used to simulate virtual users and measure how well an application handles various loads. It helps identify performance bottlenecks by generating load on the application and analyzing its response times and behavior under stress.
6. LoadUI Pro: LoadUI Pro is a commercial load testing tool designed for testing the performance and scalability of web applications and APIs. It enables users to simulate various load conditions, monitor system behavior, and identify performance bottlenecks to ensure applications can handle real-world usage.
7. LoadView: It is a cloud-based performance testing tool used to simulate high traffic and load on websites and applications. It helps identify how a site performs under various conditions by generating real user interactions and analyzing the impact on performance.

Advantages of Load Testing

Load testing has several advantages that make it an important aspect of software testing:

1. Identifying bottlenecks: Load 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.

2. Improved scalability: By identifying the system's maximum capacity, load 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.
3. Improved reliability: Load 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.
4. Reduced risk: By identifying potential issues before deployment, load testing helps reduce the risk of system failure or poor performance in production.
5. Cost-effective: Load 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.
6. Improved user experience: By identifying and addressing bottlenecks, load testing helps ensure that users have a positive experience when using the system. This can help improve customer satisfaction and loyalty.

Disadvantages of Load Testing

To perform load testing there in need of programming knowledge. Load testing tools can be costly. Load testing also has some disadvantages, which include:

1. Resource-intensive: Load testing can be resource-intensive, requiring significant hardware and software resources to simulate a large number of users or transactions. This can make load testing expensive and time-consuming.
2. Complexity: Load 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 load testing.
3. Limited testing scope: Load 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 load testing with other types of testing such as functional testing, regression testing, and acceptance testing.

4. Inaccurate results: If the load testing environment is not representative of the production environment or the load test scenarios do not accurately simulate real-world usage, the results of the test may not be accurate.
5. 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.
6. Complexity in analyzing the results: Load testing generates a large amount of data, and it can be difficult to analyze the results and determine the root cause of performance issues.