Performance Testing for an eCommerce Store

Client Overview

The Client is a prominent online retailer with a customer base of over 50,000 active users. They specialise in selling a wide range of products, including clothing, accessories, and electronics, through their eCommerce platform.

Business Requirement

The Client's primary objective was to assess the performance and capacity of their eCommerce platform under heavy load conditions. Specifically, they aimed to determine the maximum number of concurrent users the system could handle effectively. The scope of the project included:

I. Conducting real-time load tests to simulate a high number of concurrent users (at least 5,000 users) accessing the eCommerce platform.

II. Monitoring and recording the system's stability and response time during the load testing phase.

III. Validating the core functionality of the platform to ensure it met minimum performance standards.

IV. Detecting instances of server crashes, incorrect processing, and slow responsiveness under various load test scenarios.

Engagement Objectives

The engagement focused on achieving the following key objectives:

I. **Core Functionality Validation:** The primary aim of the performance testing was to verify that the core functionalities of the eCommerce platform were capable of meeting the established minimum performance standards. This involved assessing critical processes such as user registration, product browsing, cart management, checkout, and order processing.

II. **Detection of Server Crashes and Processing Issues:** The load testing aimed to identify potential vulnerabilities in the system, including instances of server crashes, incorrect processing of user requests, and delayed response times. By simulating a heavy user load, the testing process sought to uncover weaknesses that could lead to system failures or customer dissatisfaction.

III. **Assessment of High Parallel Load Impact:** Another significant goal was to uncover any problems that might arise when the system experienced high parallel loads. The testing process would help identify bottlenecks, resource limitations, and performance degradation that could occur when numerous users accessed the platform simultaneously.

Tool Stack

These tools collectively enabled the creation, execution, monitoring, and analysis of performance tests, yielding valuable insights into the system's behavior, scalability, and responsiveness under various load conditions.

I. **Apache JMeter:** Apache JMeter is an open-source, Java-based load testing tool widely used for performance testing, functional testing, and stress testing of web applications, APIs, and other software. Apache JMeter was leveraged to automate API calls and record functional scenarios. Its scripting and test building capabilities allowed for the simulation of user actions and load scenarios, providing valuable insights into the system's performance under various conditions.

II. **BlazeMeter Taurus:** BlazeMeter Taurus is an open-source test automation tool that extends and simplifies the functionality of JMeter. It provides an abstraction layer that simplifies the creation, execution, and analysis of performance tests. Taurus enables users to define test scenarios using a YAML or JSON file, making it easier to manage complex test configurations.

III. **APDEX (Application Performance Index):** APDEX is a standardized metric used to measure user satisfaction with the response time of applications or services. It provides a numerical score that quantifies user experience, taking into account response time thresholds for satisfied, tolerating, and frustrated users. In this case study, JMeter's APDEX tool was utilized to generate performance test result reports, offering a clear measure of user satisfaction and helping to set performance benchmarks.

IV. **Perf-mon Plugin:** The Perf-mon plugin is an extension for JMeter that enables the monitoring of system-level metrics such as CPU utilization, memory consumption, disk I/O, and network activity during load testing. This plugin was employed to monitor the CPU and memory utilization of application and database servers, providing insights into system resource consumption under varying load conditions.

V. **Amazon EC2 (Elastic Compute Cloud):** Amazon EC2 instances were used as distributed load generators for the performance test cycles. EC2 instances provided scalable and on-demand computing resources to simulate a high number of concurrent users accessing the eCommerce platform.

Our Solution - Performance Analysis Principles for the eCommerce Web Application

In addressing the performance testing objectives for the eCommerce web application, we employed a comprehensive approach based on three crucial principles: Speed, Scalability, and Stability. By systematically evaluating these aspects, we were able to gain valuable insights into the system's performance and readiness under varying load conditions.

1. Speed: Speed is a critical factor that significantly influences user satisfaction and engagement on any web application. To assess the speed of the eCommerce platform, we focused on key functions and measured their load times, including the crucial "first-byte time." This involved analyzing the time it takes for the server to respond to a user request, providing an initial indication of the system's responsiveness.

Key Actions Taken:

- Identified critical functions within the eCommerce application, such as product browsing, cart management, checkout, and order processing.
- Developed performance tests to simulate user interactions with these functions under different load levels.
- Measured and analyzed load times, response times, and first-byte times to determine the application's speed performance.

2. Scalability: Scalability is paramount for a web application's ability to accommodate increasing user demand without compromising performance. To evaluate the platform's scalability, we employed automated API scripts to determine the maximum load that the web servers could effectively handle. This helped us understand the system's capacity to scale horizontally under high user loads.

Key Actions Taken:

- Created automated API scripts to simulate a gradually increasing user load on the web servers.
- Monitored system metrics such as CPU utilization, memory consumption, and network activity as the load increased.
- Identified performance bottlenecks and resource limitations that could affect the application's ability to scale.

3. Stability: Stability is a critical indicator of an application's resilience and reliability under varying load conditions. We utilized the same API scripts used for scalability testing to assess whether the eCommerce web app had stabilized under different load levels. Stability testing helped us determine if the application exhibited consistent performance and maintained acceptable response times under sustained load.

Key Actions Taken:

- Ran performance tests with the automated API scripts to apply varying levels of load to the system.
- Monitored system behavior and performance metrics during the testing period.
- Analyzed the application's responsiveness, error rates, and resource consumption to ascertain its stability under different loads.

Approach and Execution Strategy for Performance Testing

The successful execution of performance testing requires a well-defined approach that encompasses understanding, planning, script creation, execution, and monitoring. Our approach follows a systematic process to ensure accurate and comprehensive performance assessment of the eCommerce web application.

1. Understanding the Context: Before initiating performance testing, it's crucial to gain a deep understanding of the web application and its usage patterns. For existing websites or updates, we leverage web analytics data to identify traffic patterns, user behavior, and peak usage times. This helps us establish a baseline for performance testing scenarios and load profiles.

2. Test Planning: Based on the insights gained from the initial analysis, we create a comprehensive test plan for performance testing. This plan outlines the objectives, scope, testing environment setup, load scenarios, user behavior simulation, performance metrics, and monitoring strategy.

3. JMeter Script Creation: The next step involves the creation of JMeter test scripts. These scripts simulate user interactions and load scenarios on the eCommerce platform. We design test scripts that emulate typical user behavior, such as browsing products, adding items to the cart, and completing the checkout process. The scripts are designed to reflect various usage patterns and load levels.

4. Test Execution: Once the JMeter test scripts are developed, our team executes the tests using the Apache JMeter software. These tests simulate real-world user activities by generating HTTP requests and measuring response times. The tests are conducted under different load levels, gradually increasing the number of concurrent users to evaluate the application's performance limits.

5. Performance Monitoring: To gain a comprehensive view of system performance, we integrate performance monitoring tools. These tools track and measure various metrics, including CPU utilization of both web and database servers. Monitoring tools provide real-time insights into resource consumption, allowing us to identify potential bottlenecks, resource limitations, or performance degradation.

6. Analysis and Optimization: The data collected during test execution is meticulously analyzed to assess the application's performance characteristics. We identify any performance bottlenecks, areas of concern, or deviations from expected

behavior. This analysis informs optimization strategies and recommendations to enhance the application's speed, scalability, and stability.

7. Reporting and Recommendations: A detailed performance test report is generated, summarizing the test objectives, methodology, results, and insights. The report highlights key performance metrics, such as response times, throughput, error rates, and CPU utilization. Additionally, the report provides actionable recommendations for improving performance, addressing identified issues, and ensuring a seamless user experience.

Load/Stress Test Types and Schedules

In our performance testing strategy for the eCommerce web application, we conducted a series of tests using the provided APIs to assess its capacity, resilience, and performance under different conditions. Each test type served a specific purpose and provided valuable insights into the application's behavior. The following are the load/stress test types that were executed, along with their schedules:

1. Capacity Test: Objective: To determine the maximum number of concurrent users that the application server can support while maintaining an acceptable response time and error rate.

Test Schedule: Conducted during a period of anticipated peak usage, such as a major sale or promotional event.

Key Actions:

- Gradually increased the number of concurrent users accessing the application.
- Monitored response times, error rates, and server resource utilization.
- Identified the point at which the system's performance started to degrade and error rates increased.

2. Single Function Stress Test: Objective: To assess the application's reaction to extreme load on a specific function with no wait times and no ramp-up time.

Test Schedule: Conducted as a focused test after the capacity test or during planned maintenance.

Key Actions:

- Simulated a large number of users simultaneously performing a single specific function (e.g., adding items to the cart).
- Measured how the application's performance and response times were affected under this intense load on a narrow area of the code.

3. Consistent Load Test: Objective: To evaluate the application's ability to sustain acceptable performance levels over an extended period (at least 6 hours) under a continuous load.

Test Schedule: Conducted during non-peak hours to minimize impact on users.

Key Actions:

- Applied a consistent load to the application server over an extended period.

- Monitored performance metrics, server resources, and any signs of performance degradation.
- Aimed to identify potential memory leaks, resource exhaustion, or other issues that might arise over time.

Test Deliverables for Performance Testing

Our performance testing effort for the eCommerce web application was concluded with a set of comprehensive test deliverables. These documents and artifacts provide a clear overview of the testing process, results, and insights, enabling the Client to make informed decisions and improvements. The following are the key test deliverables provided:

1. Performance Test Plan: The Performance Test Plan is a detailed document outlining the entire scope, objectives, methodology, and approach of the performance testing effort. It includes information about test objectives, load scenarios, user profiles, testing environment setup, testing tools, success criteria, and monitoring strategy.
2. Test Scripts: The Apache JMeter test scripts were developed based on the discussions with the client and cover all major functionalities of the eCommerce web application. These scripts simulate user interactions and load scenarios, accurately reflecting real-world usage patterns.
3. Test Results Data: The Test Results Data comprise the raw data collected during the performance test runs. This data includes metrics such as response times, throughput, error rates, and server resource utilization. It serves as a comprehensive record of the application's behavior under different load conditions.
4. Test Results Final Report: The Test Results Final Report provides an in-depth analysis of the performance tests conducted according to the test plan. It offers a comprehensive overview of the application's performance, highlighting key metrics and insights. The report includes details such as:

- Overall system speed during and after load bursts.
- Incidents of timeouts or non-HTTP response codes during load tests.
- Minimum, Maximum, and Average response times of the system.
- Throughput and latency recorded for each API endpoint during tests.
- Response codes and messages received from the server.
- Application and database server CPU/memory utilization statistics.
- Network utilization statistics based on AWS console access.

Impact of Our Solution on the eCommerce Store's Performance

Our rigorous performance testing efforts and detailed root cause analyses yielded significant improvements in the performance of the eCommerce store. Through our solution, we were able to identify, address, and provide recommendations for the

issues that were affecting the store's speed, stability, and scalability. The impact of our solution can be summarized as follows:

1. Pinpointing Actual Issues: Our performance testing efforts, along with accurate Root Cause Analyses (RCAs), allowed us to precisely identify the underlying issues that were impacting the store's performance. By thoroughly analyzing the test results and system behavior, we provided the client with actionable insights into the specific areas that required attention.
2. Educating Development and Deployment Teams: We actively engaged with the client's development and deployment teams to educate them about the performance issues identified. Our collaboration helped the client's teams understand the root causes of the problems and guided them in implementing effective solutions.
3. Providing Recommendations: We offered targeted and practical recommendations to the client on how to address the identified issues. These suggestions aimed at improving the store's speed, stability, and scalability, ensuring a better overall user experience.
4. Iterative Testing and Reporting: Our approach involved multiple iterations of performance tests, with separate test reports provided for each iteration. This iterative process allowed us to track progress, measure improvements, and validate the effectiveness of implemented solutions. The detailed test reports presented data in both tabular and graphical formats, offering a clear and comprehensive view of the issues and their resolutions.
5. Achieving Performance Goals: Through collaborative efforts, we achieved significant performance enhancements, meeting the following performance goals:

- **Average Response Time:** Reduced to less than 5 seconds (measured by Time to Last Byte metric).
- **Maximum Response Time:** Reduced to less than 30 seconds (measured by Time to Last Byte metric).
- **Database Server CPU Utilization:** Maintained at less than 60% under maximum stress.
- **Application Server CPU Utilization:** Maintained at less than 70% under maximum stress.
- **Server Errors:** Ensured that the maximum number of acceptable server errors (non-HTTP-200 status codes) on client requests were less than 3% of all client requests.

Conclusion

Through our comprehensive and collaborative approach to performance testing, we successfully addressed the performance challenges faced by the eCommerce store. By identifying and rectifying bottlenecks, educating the client's teams, and providing actionable recommendations, we significantly improved the store's speed, stability, and scalability. The achievement of performance goals and the positive impact on

the eCommerce platform's overall user experience demonstrated the effectiveness of our solution in optimizing the client's online store for peak performance.