Performance Testing Micro-Credential

Syllabus

Table of Contents

Performance Testing

References

General Information

STUDY TIME – 200 MINS.

KEYWORDS

concurrency testing, load test, objectives-based reporting, operational profiles, performance testing, risk-based reporting, scalability, stress test

LEARNING OBJECTIVES FOR PERFORMANCE TESTING

Introduction

The Purpose of Performance Testing

(K1) Recall the goals of performance testing
(K2) Explain why defining and getting agreement on performance requirements is important
(K2) Explain how performance testing aligns with the SDLC

Performance Testing Risks, Benefits, and Challenges

(K1) Recall the unique challenges of performance testing
(K1) Recall the risks of performance testing
(K1) Recall the benefits of performance testing

Performance Testing Approach

(K2) Describe the components that should be included in a performance test plan
(K2) Explain the factors to be considered when defining the test approach

Conducting Performance Testing

(K1) Recall the steps for conducting performance testing
(K1) Recall the components that should be checked during performance test preparation
(K1) Recall how performance test results differ from functional test results
(K1) Recall the types of performance test reporting

Performance Test and Analysis Tools

(K1) Recall why tools are necessary for performance testing
(K2) Describe the challenges that can be encountered with performance testing tools

Introduction

Performance is a major quality factor in most systems and applications, regardless of the computing environment. A system or application may be functionally correct, but if it fails to deliver the needed performance it is likely to be considered a failure. Performance testing is one way to measure system performance in advance of deploying the system.

Performance testing should start at the component (unit) test level and continue until system deployment. Waiting until the end of a project to conduct performance testing carries the high risk that performance problems will not be solvable due to time, money and technological constraints.

One of the most costly and publicized performance failures occurred during the Facebook Initial Public Offering (IPO), when the NASDAQ stock exchange could not handle the volume of trading on the day of the IPO. The cause of the failure was determined to be three lines of code that got into an endless loop. To date, NASDAQ has paid over $80 million in fines and restitution. In addition, the United States Securities and Exchange Commission imposed requirements on NASDAQ to help prevent similar failures in the future [NASDAQ].

The purpose of Performance Testing:

Performance testing can have multiple goals and purposes, including:

Measuring system performance under given conditions

Determining the maximum concurrent user load or transaction volume a system can handle before it fails

Providing information to assist in capacity planning for a system

Significant time and money can be invested in performance testing. In order to obtain the best return on investment, the testing must be targeted correctly and the goals clearly defined. This can be difficult since different stakeholders may have varying, even conflicting, views of “acceptable” performance. In order to be successful, there must be agreement in the project team regarding what the performance testing will measure and what will constitute a successful result.

Defining Performance Requirements and Getting Stakeholder Agreement

Documenting performance requirements can be challenging. Eliciting the requirements can be even more difficult. When documented, performance requirements are normally recorded in over-arching requirements, such as “all system responses to users that require longer than 3 seconds must display a wait notification”. In an Agile environment, these requirements may be documented in specific non-functional Epics.

Without stakeholder performance requirements, it is hard to know if the observed levels of performance are adequate, making it difficult for testers to make a reasoned assessment of test results. A complicating factor in determining system performance requirements is that the assessment of “acceptable” performance can be subjective.

Getting agreement on performance requirements can be quite challenging because of the subjectivity of opinions and the cost required to achieve higher levels of performance. For example, one stakeholder group might feel that a response time of two seconds is acceptable, while another group might feel that a response time of one second or less is required. The cost to achieve the one second response time might be quite high (e.g., 5x improvements in hardware, more robust networks, major modifications to code or databases), whereas two second response time may be easily achieved.

Aligning Performance Testing in the SDLC

As with all forms of testing, performance testing needs to be integrated with the SDLC. Regardless of the SDLC, full lifecycle performance testing is needed to detect performance problems early, when they are easiest to fix.

The following activities in an SDLC have key tasks for performance testing:

- During requirements definition, the specific requirements for performance (and definition of the expected load) should be clearly defined and agreed to by all stakeholders. This will eliminate debate when the results are reviewed later.
- During design and coding, performance factors must be considered and built into the design and the subsequent code. Inefficient code or an inefficient design will be difficult and costly to fix later.
- During integration and system testing, new performance testing opportunities can appear. Any new integration may introduce inefficiency and potential performance bottlenecks.
- During acceptance testing, a major objective is to assess performance within the business or operational context. In some cases, performance testing is part of contract acceptance testing and operational acceptance testing.

Risks, Benefits, and Challenges

Performance testing has unique challenges, primarily due to lack of understanding of the complexity and cost of good performance testing.

These include the following:

- Getting definition and agreement from stakeholders with regards to acceptable system performance
- Getting adequate funding for performance testing tools
- Acquiring the best fit solution for a performance testing tool, within the schedule and budget of the project
- Accurately profiling load levels at given periods of system usage
- Acquiring skilled test engineers to plan, design and conduct a realistic performance test
- Building a representative performance test environment

Unless these challenges are met and understood, the performance test effort may never start or will not be effective. In addition to the above challenges, performance test efforts have risks that must be considered and mitigated in order to achieve the expected benefits.

Risks

Performance testing has the following risks:

- Inadequate performance test design, resulting in inaccurate or incomplete test results
- Incomplete coverage of protocols and connectivity, resulting in important aspects of performance testing for a particular system or application being missed
- Inadequate test environments, resulting in inaccurate test results and erroneous conclusions
- Inability to apply the performance tool correctly, resulting in inefficient, inaccurate, and inconclusive tests

- Inadequate coverage of functions, resulting in incomplete test results
- Inadequate amounts of user and data load, resulting in inaccurate and incomplete test results
- Lack of defined stakeholder requirements for performance, resulting in the inability to identify performance targets
- Lack of agreement on defined stakeholder performance requirements, leading to conflicts concerning acceptable levels of system performance
- Lack of appropriate performance metrics, resulting in test reports that are incomplete or lack depth of meaning.

Benefits

If the challenges are met and risks mitigated, significant benefits can be expected from performance testing.

These include the following:

- Opportunities to "right-size" the system to handle expected load
- Opportunities to plan mitigation steps if loads exceed expected levels
- Early test results can help define the acceptable levels of performance
- Expectations for future growth can be tested to know if the system will be able to support it
- System performance weaknesses can be identified and fixed before being discovered in production
- Long lead time items, such as adding more hardware or improving network resiliency, can be addressed in adequate time before a production launch

Because performance issues may be expensive or time-consuming to rectify, identifying them early on allows the greatest chance for mitigation before an exposure in the production environment. Performance testing is often intended to merely confirm expectations that the overall system is fit for purpose, but frequently results in identifying significant problems that must be corrected.

Performance Testing Approach

Performance testing often requires a separate planning effort. This effort is focused on the particulars of the performance testing, such as sizing the test systems, procuring resources and planning uninterrupted testing time.

Defining the Test Plan

Performance test plans generally include the following information:

Scope

The correct setting of scope in performance testing is a critical activity to ensure that the targeted objectives are met. The scope is often framed by the following:

- **Functionality** – What features will be included or excluded from the performance testing?
- **Architecture** – Which aspects of the system architecture (e.g., networks, APIs, devices, databases) will be included?
- **Transactions** – What typical user transactions will be included in the testing?
- **Users** – Which classes of users should be included based on selection factors (e.g., location, type of transactions, demographics, concurrent usage)?
- **Data** – How much data should be used and how should it be accessed?

Strategy and Approach

The performance test strategy and approach can define the way performance testing is to be conducted. For example, an organization might decide to outsource performance testing if it lacks the tools and skilled people to conduct the testing on its own.

Risk Assessment

As in much of testing, performance testing can also be risk-driven to focus testing on areas of the system, or on functions and transactions,

that carry the highest risk, such as those with the highest use.

Definition of Test Objectives

Performance test objectives describe, at a high level, what the performance test is intended to achieve. Oftentimes, these are worded as performance objectives to verify and/or validate. For example, “Verify the system response time is 1 second or less at peak load times.”

Responsibilities

Team roles must be defined, along with roles and responsibilities for those external to the team and the organization. Performance testing often requires the support of system architects, database analysts, network engineers and other specialists who can participate in troubleshooting and system tuning.

Reporting Metrics

Metrics have a very important role in evaluating performance test results. It is important to define which metrics are most meaningful to stakeholders – both business and technical – and ensure those metrics are tracked by the selected monitoring tools. Some metrics, such as average response time, may not be meaningful or as useful as the response time for 90% of the users. This will help eliminate any outliers that can skew the average.

Defining the Test

Depending on the formality of the environment, the test approach may be defined as part of the performance test plan or defined separately. The test approach must consider the following:

Environments

Test environments are a key concern in performance testing because of the need to obtain representative results. The challenge is that full-scale dedicated test environments are often difficult to build due to cost and complexity. One possible alternative is to use a cloud-based test environment that can be rapidly scaled to simulate production configurations.

Unfortunately, it is not wise to simply predict performance at higher levels of scale based on lower system capacities. It is normally necessary to apply realistic load with realistic system conditions.

In some cases, such as in new system development, the system under development can be tested in the same system configuration that will eventually become the actual production system.

Load and Throughput Profiles

To accurately design performance tests, current and predicted profiles must be understood for both load levels and data throughput, such as peak load times and load levels. A common way to know performance profiles is to measure and study user behavior and levels as captured in system analytics.

Operational Profiles

Operational profiles describe the functions users are expected to perform on the system and the frequency of these activities. These can be seen as simple functions or wider-ranging workflows for end-to-end transactional testing of the system, sometimes called “transaction threads”. Operational profiles are implemented as scripts to be executed by virtual users, creating an accurate environment for measuring performance.

Test Data

Representative amounts and types of test data are often at the heart of performance testing. This data may be generated by tools or copied from a production environment if no private data is involved.

Conducting Performance Testing

In order to run the performance test, several other steps are needed. The tests must be prepared (including the environment and data), they must be executed, the results must be evaluated and reported, and there may be a need for on-going monitoring.

Test Preparation

Prior to test execution, the following components should be verified as ready for the testing:

Test Environment

It is often helpful to run preliminary tests to make sure basic test environment elements are in place and working correctly and that proper access is available. It is also important to validate that all co-existing applications and systems that could cause a performance load are in place and operational, as these systems can also place performance load on the test environment.

Test environment capacities need to be verified as correct. For example, CPU levels, database size and network configuration all have an impact on system performance.

Data

All databases and files should be populated with the designed volumes of test data to ensure the tests are not blocked due to insufficient or incorrect data. Generating this data and verifying its correctness must be conducted prior to performance test execution.

Testware (Test Scripts, Procedures, etc.)

As part of test preparation, it is vital that the designed and implemented testware is in place and executing correctly as designed. An effective activity is to conduct a preliminary test using a representative sampling of test procedures to verify functional correctness and data accessibility and accuracy.

Test Execution

If the preparatory steps have been completed, then test execution becomes a matter of running the tests in the tool and monitoring the execution. Monitoring features are a part of many performance test tools. However, the interpretation of test results requires human intelligence.

Sometimes, it is only by observing test results in real-time that some performance anomalies may be identified. For example, it is common in a performance test to ramp up load levels. In some test tools, the impact of increasing system load can be seen by analyzing graphical representations after the test is complete. However, other tools may primarily display test results in a text-based format. In those cases, it is helpful to observe system behavior and metrics as the test is in progress.

Test Evaluation

In contrast to functional testing, where test outcomes can be evaluated as “pass” or “fail”, performance test results tend to be more comprehensive and informative. The main question that performance testing seeks to answer is “Does system or application performance meet stated performance goals or requirements?”

It is important to understand that performance testing is highly dynamic and dependent on many factors, such as user load, workflows performed, system capacity (CPU speed, database efficiency, code efficiency, and networking configuration), and system configuration. Changing any of the variables can significantly affect the results.

Performance testing is a snapshot view. Even a small change to the system can cause improvements or degradation of performance. For this reason, performance testing should be performed throughout system development and after release to production.

Performance testing is sometimes used to assess the system’s capabilities. While load testing can determine performance levels at given load levels, stress testing can reveal the maximum capacity the system can handle based on a given configuration. This information can help capacity planners to know if system upgrades will be needed, and if so, when they may be needed.

Test Reporting

The main deliverable of performance testing is the reporting of the test results. Performance test reporting will vary depending on the audience and the purpose of the test. Since each stakeholder group has differing levels of technical understanding, test results need to be presented in ways that each group can understand and find meaningful.

Risk-based Reporting

As in other forms of testing, risk-based reporting can help identify which aspects of system performance may carry the most relative risk. This risk-based view of performance test reporting can help stakeholders understand where to focus efforts for the most effective system implementation decisions.

Objectives-based Reporting

In objectives-based reporting, performance test results are reported with traceability to performance test objectives, such as the expected response time to a specific user action. This allows testers and stakeholders to know whether or not performance objectives have been met as demonstrated by the success or failure of performance tests.

Performance Test and Analysis Tools

Why Tools are Essential for Performance Testing

Performance testing is one type of testing that is not feasible without tools.

Tools are used for the following:

- **Creating simulated concurrent user load** – This simulated load is used for concurrency testing. It is more precise and does not require as many resources as trying to test with large numbers of people. Concurrency testing is used to determine how the system will perform with a consistent load of virtual users who are "concurrently" using the system.
- **Sustaining high levels of load** – Even if it was possible to generate enough load with actual people, it would be difficult to sustain the load long enough to get a good measurement or to allow repetition of the test.
- **Accurately measuring load levels and system response time, CPU utilization, memory allocation, etc.** – It takes more than a stopwatch to measure response times. Most performance test tools have features to measure aspects of the SUT that directly impact performance. For tools that lack robust monitoring functionality, it is possible to obtain tools that independently monitor test performance.
- **Repeating performance testing whenever needed** – Performance tests must be repeated as changes are made to the application and system. Often, during performance testing, system tuning takes place to improve the performance. This requires re-execution of the tests to ensure that the expected improvement has been realized.

Tool Challenges

Although tools are needed for performance testing, there are some challenges to overcome:

- **Performance test tools do not know what to test –** The performance tester must determine which functions to test, how many concurrent users to simulate, which data to use, and other conditions to achieve the test objectives.
- **The environment has to be representative –** For many people, building a performance test environment may be the greatest challenge of all. Adequate performance test environments may require an investment in hardware, software, people, and tools – all scalable to the level of production use. Cloud- based virtual test environments have become an attractive alternative to physical environments in some cases.
- **High volumes of test data will be needed –** This may require the use of test data generation tools or strategies to modify a copy of existing production data.
- **Expertise is needed –** Performance testing requires skills and experience not commonly found in many organizations. It is common to obtain outside consultation when first starting to plan and conduct performance testing.
- **Tools can be expensive –** The most robust and popular performance test tools can be very expensive. While new tools in the marketplace help to keep pricing competitive, organizations that have large investments in performance test tools may be less inclined to switch to more affordable tools. Even open source tools have a cost in learning and maintenance.

Selecting the proper tools requires evaluating a number of factors including long-term cost, training requirements, vendor reliability, etc. Because performance tools may be significantly expensive, it is important to consider the life expectancy of the tool, the ROI and the likely future requirements for tooling.

References

ISO/IEC/IEEE Standards

- ISO/IEC/IEEE 12207:2017
- ISO/IEC/IEEE 15288

Trademarks

The following registered trademarks and service marks are used in this document:

- AT*SQA® is a registered trademark of the Association for Testing and Software Quality Assurance

Books

[Anderson00]: Anderson, L.W. and Krathwohl, D.R. (2000) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon: Boston MA, ISBN-10: 080131903X

[Firtman]: Maximiliano Firtman, "Programming the Mobile Web", O'Reilly Media;

Second Edition (April 8, 2013), ISBN-10: 1449334970

[PMBOK] Project Management Institute, "A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Sixth Edition, 2017, ISBN-10: 9781628251845

Other References

The following references point to information available on the Internet. Even though these references were checked at the time of publication of this syllabus, AT*SQA cannot be held responsible if the references are not available anymore. AT*SQA is not endorsing any of these sites or their products. The references are provided as a source of information only.

https://techcrunch.com/2013/03/25/ip-oh-my-gosh-all-that-money-just-disappeared/ https://www.reuters.com/article/us-facebook-settlement/facebook-settles-lawsuit-over-

2012-ipo-for-35-million-idUSKCN1GA2JR

[NASDAQ] https://www.sec.gov/news/press-release/2013-2013-95htm

National Institute of Standards and Technology. Framework for Improving Critical Infrastructure Cybersecurity. Version 1.1. 2018. https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf

National Institute of Standards and Technology. Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy. Revision 2. 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf

[WCAG] https://www.w3.org/WAI/policies/

www.atsqa.org