Performance & Load Testing AEM - Part 1
Author Basics

Tim Donovan, Technical Architect, Adobe Partner Experience

Introduction

Before every go live, time should be spent ensuring that the AEM application developed will perform adequately, not only against the intended number of website visitors but against the number of authors and authoring operations executing on it.

This series will explore the topic of testing and the basic best practices to ensure that you perform adequate AEM author load and performance testing. Future papers will discuss in further detail how tools like ToughDay and JMeter can be utilized, and give practical examples of testing scripts.

Purpose of Testing

Before starting to develop a load and performance testing plan, the purpose of the tests should be clearly defined. The purpose usually falls into one or more of the following categories:

Longevity test

Run the standard set of authoring action tests for a long period (four hours at a minimum but ideally for 4-6 days per AMS best practices) that emulates normal system load. This will expose issues with resources, such as memory leaks.

Identify maximum system capacity

If a system is likely to see expanded use (for example, through onboarding new business units), an estimate for the breaking point of the system, while performing standard authoring tasks, should be found. This is typically measured in the number of authors; thus the solution is to ramp up the threads (users) performing standard authoring activities until the throughput drops (usually it noticeably drops at a certain point, so it is obvious) or until one of the activities falls below an agreed KPI.

KPI's check under normal load

Run the standard set of authoring action tests, under normal system load, for a shorter period and check them off against the KPIs.

KPI's check for drops in performance

A more advanced test, but this entails running the standard set of authoring action tests, under a normal system load while in parallel running each authoring action individually and examining its effect on the other authoring tests.

Ideally, all the above-mentioned tests should be performed, with project time and budget allocated

accordingly. With limited time and budget, the most practical value can be obtained from the longevity test, as a basic and manual check of system KPI's can be performed in parallel. It is also best practice (especially within the context of an Agile project) to run performance testing as part of a Continuous Integration (CI) process.

Requirements for Testing

Environments

The Adobe documentation on *Best Practices for Performance Testing*[1] lists this, due to its importance, as the first topic. Without testing on an environment that is an accurate representation of the production system or without putting the environment under a load that is similar to the load faced in production, the testing is largely invalidated by meaningless results. If the AEM solution is going to integrate with other systems, these systems also need to be present.

Testing Scripts

Again, simulating reality is the key to useful tests; therefore, a test script that executes *standard authoring activities* on the Author instance is necessary. These will make up the JMeter test plan and be composed of individual actions to roughly mimic a user's behavior. These will be different in every environment, but are often at a minimum:

- Login
- Navigating the UI
- Open a page
- Create a new page
- Adding components
- Adding assets
- Uploading assets
- Publishing page
- Page rollout
- Reference search

The easiest, but not 'best practice' approach, is to build a test that calls a web API and measures the response time. For example, for performance testing page creation in AEM, you could simply call and time the response of the following command:

/bin/wcmcommand -cmd='createPage'

However, this does not simulate the actual calls made when a content author follows the new page wizard steps in AEM. Rather than a single call, AEM makes around 80 requests for various resources, and if these are not included in the test script, you will *not* be close to simulating reality. It is recommended that you analyze the network tab in a browser to see what communication occurs or use a test script recorder that records real user traffic through a proxy to make test scripts. A delay between each action is also recommended and using a standard throughput timer, rather than a simple timer, is a good way to introduce a realistic pause between tests.

[1] https://docs.adobe.com/docs/en/aem/6-2/deploy/best-practices/best-practices-for-performance-testing.html

Definition of Normal System Load

For testing to simulate the likely load on the system, the business will need to define what the normal load is likely to be. To do this, estimate how often, per hour, the various authoring activities take place. It is likely this figure will need adjusting during testing, but if the solution is already (prematurely) live, the access logs can be used to deduce the actual load. Note, the Adobe best practice is to test at 150% of expected system load. For example, if the business says there are ten publish operations an hour, test with 15 operations to allow for unexpected system peaks.

Key Performance Indicators

Agree on indicators

KPIs for system operations need to be defined and agreed upon with the client. These are not usually defined for smaller and granular authoring operations, such as how long opening a component dialogue takes, but are on operations such as such page publishing or a login operation. That is not to say we cannot measure these, but it requires more time and investment to plan and measure. If a smaller and granular operation is slow, usually poor system performance will be noticed across all system tests.

Define a target value

There are two ways to achieve this—the first is to measure the performance of an operation directly on the system and use this performance value as a target for the future tests under higher load. The second is to define a tentative goal, or a desirable value, based on user's expectations and feedback. Both are valid approaches and may need refinement as testing begins.

Interpretation of results

The performance measurement is done across multiple executions of transactions by the simulated concurrent users and across multiple performance tests. These samples must be handled from a statistical point of view. You should choose which statistics will be considered as a mean value in your case.

It is recommended to choose the median value (aka 50th-percentile) as it represents the user experience in most cases. Once this value is stable, the focus can be set to the worst experience usually represented by the 90th or 95th percentile. The following is recommended reading on the subject: https://www.dynatrace.com/blog/why-averages-suck-and-percentiles-are-great/

Knowing the standard deviation of your test result data is valuable and is usually available as an output from a test plan. The smaller the deviation, the more consistent the data, and one wants to see a low variation over repeated tests. If the standard variation is high, it shows that something is wrong; unless there is an issue with the tests performed (for example, the deviation on test operations performing different queries is invalid, as different queries naturally result in execution time variation), it demonstrates the performance of the system cannot be accurately guaranteed.

Further Reading

Additional topics in this series will include:

- Testing Author – Advanced: Exploring JMeter in Detail
- Testing Publisher – Basics: Best Practices
- Testing Publisher – Advanced: Exploring JMeter in Detail