Performance Testing Mentoring Program 2022-2023

Task 12 Part 1

Created by: Serhii Diedov

Date: May 2023

BlogEngine.NET 3.2 PERFORMANCE TESTING STRATEGY

1 Introduction

BlogEngine .NET 3.2 (System Under Test, SUT) is a web application designed to create and administrate a blog, implemented in two data source options namely file system and database. The purpose of testing determined by the customer is to determine with which data source the server side of the system is more productive and stable, including with an increase in data volume, and also has the best potential for scaling.

This document will outline the scenarios, tests, parameters and data used to achieve the above goal. The scope of tests described in current document is to verify how SUT behaves under different load types. All deliverables should be used for internal (in-house) usage only and should not be presented to public.

This document describes the strategy of performance testing for the BlogEngine .NET 3.2 project. It consists of outlines for the following items:

• Scope of testing, test descriptions

• Non-functional requirements (NFR) related to performance

• Approach

• Main scenarios

• Test data

• Requirements for test environment

2 Items to be tested

BlogEngine .NET 3.2 application will be tested via UI with main focus on the server side. Testing the client side in agreement with the customer is taken out of scope.

In the application, users have 3 main roles "Admin", "Editor" and "Anonymous" user. Below is a list of functionality with an indication of the roles under which this functionality will be used in test scenarios.

#	Page Name/Feature	Roles
1.	Open Home page	Anonymous
2.	Open Contacts	Anonymous
3.	Open Large Calendar	Anonymous
4.	Open Predefined Date	Anonymous
5.	Open Random Date	Anonymous
6.	Open Random page =1	Anonymous
7.	Open Random page >1	Anonymous
8.	Search by Name	Anonymous
9.	leave a comment	Anonymous
10.	Open first post	Anonymous
11.	Open Random post	Anonymous
12.	Open Home page	Anonymous
13.	Open login page	Admin
14.	Login	Admin
15.	Open Admin Page	Admin
16.	Open USERS menu	Admin
17.	Open Users page	Admin
18.	Add user	Admin
19.	Delete user	Admin
20.	LogOff	Admin
21.	Login	Editor
22.	Open editpost page	Editor
23.	Open Predefined Date	Editor
24.	Open Random post	Editor
25.	Editpost	Editor
26.	LogOff	Editor

High-level description of test scenarios: simulating user behavior on the site, where:
Admin - Open home page, log in as Admin user, open Admin's page, check number of Anonymous users and decide to add or delete user, finally Log out

Editor - Open home page, log in as an Editor user, open a predefined date, open a random post and edit it, rapid it 50 times, and finally Log out.

Anonymous user - visits different pages, opens posts and leave comments.

3 Items not to be tested

The functionality of the whole application

4 Approach

To determine with which data source the server side BlogEngine .NET 3.2 application is more productive and stable, including with an increase in data volume and also has the best potential for scaling, a series of tests will be performed below.

First, tests will be carried out for the BlogEngine .NET 3.2 (web) for which the data source will be the file system. After that, data source from file system to DB will be switched and check the impact of changes on the system. The results for both data sources will be presented for comparison.

<u>4.1. Test types assumed for conducting:</u>

<u>Before each test:</u>

Smoke testing should be performed every time when functionality of the application and the script needs to be checked. Also, if needed can be used as a warming-up test before the main testing step.

<u>Tests to be carried out for both types of system data sources:</u>

1 Perform capacity testing (for 1000 text posts) and compare results with both file system and DB data source.

2 Perform scalability testing (for 1000 text posts) and compare results with both file system and DB data source using the following approach:

2.1 Approach to determine the impact of the number of CPUs on system performance:

a) Set the size of the memory as bigger as possible for the virtual machine (6Gb).

b) CPU scaling and perform load testing under regular load (~70% of the system capacity) for different numbers of CPUs: 1, 2, 3, 4, 6.

2.2 Approach to determine the impact of the RAM on system performance:

a) Set the number of CPUs as big as possible (6).

b) Scaling RAM and perform load testing under regular load (~70% of the system capacity) for different sizes of RAM: 2Gb, 3Gb, 4Gb (if possible), 6Gb (if possible).

3 Perform volume testing and compare results with both file system and DB data source using the following approach:

a) Testing under regular load (~70% of the system capacity) with different number of the text posts 100, 1000, 2000, 5000.

b) Testing under regular load (~70% of the system capacity)with media information and compare results for 1000 text posts and 1000 posts with a text and attached 1Mb photo

4 To ensure system stability under long-term load, **additionally perform long-time testing** under low load (~30% of the system capacity)

<u>**!!!TBD!!!** Given the time constraints, these tests can only be carried out on a system with a data source that has shown the best results in terms of stability and potential for scaling:</u>

5 Perform regular load testing (~70% of the system capacity) with the parameters based on the results of capacity testing (for 1000 text posts) **and perform stress testing** with a load of 120%, 200%, 300% and 500% system capacity. The goal is to determine the ability of the system to cope with the stress load and to identify the consequences of such a load on the system.

<u>4.2. Level of testing</u>

As the development of the application is completed, end-to-end system-level scenarios are expected to be developed with test data as close as possible to the expected data of real users.

5 Non-Functional requirements

Non-functional requirements (NFRs) are expected to be received from the customer before the completion of the test cases.

6 Test environment(s)

Considering that the resources for getting close to production's test environment are not budgeted, and the purpose of testing is to determine with which data source the server side of the system is more productive and stable, including with an increase in data volume, and also has the best potential for scaling as agreed with the customer, the following will be used test environment.

Test application and load generator are running on the same computer. Application is executed on the virtual machine and the load generator is executed on the host.

6.1. Parameters of the system under test:

BlogEngine.NET 3.2 (web) (Executed on the application server)

VirtualBox Graphical User Interface Version 7.0.4 r154605 (Qt5.15.2), Windows Server 2016 Standard (64-bit)

OS Name: Microsoft Windows 10 Enterprise, (64-bit) Version 10.0.19044 Build 19044

Base memory: 2,0Gb

Processor: Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz, 2304 Mhz, 1 Core(s), 1 Logical Processor(s)

Storage: Virtual size 60.00 GB

Network: Bridged adapter, Intel® Wi-Fi 6 AX201 160Mhz

6.2. Load generator system options

Load generator tool: Apache JMeter 5.5 (Executed on Host)

OS Name: Microsoft Windows 10 Enterprise (64-bit) Version 10.0.19045 Build 19045

Base memory: 32,0 GB

Processor: Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz, 2304 Mhz, 4 Core(s), 8 Logical Processor(s)

Storage: C: 248 Gb, D: Locked

Running Tests: Jenkins

Collection and visualization of metrics: InfluxDB, Telegraf, Grafana

6.3. Profiling tools

TBD

7 Test data

To have whole cycle of performance testing test data for BlogEngine.NET 3.2 (web) should be:

1) Reusable
2) Generated in necessary amount for different stage of testing at any time.
3) Cleanable (for example posts, comments)

1. Performance Entry, Exit, and Suspension Criteria

8.1. Entry Criteria

- Test plan is complete and approved by client.
- Correct version BlogEngine.NET 3.2 (web) is installed in performance testing environment.
- Test data is complete and in the performance testing environment in sufficient time to allow test scripts to be completed.
- Test accounts have been created in the performance testing environment in sufficient time to allow test scripts to be completed.
- Test scripts complete.
- Load generator tool and tools for collection and visualization of metrics (InfluxDB, Telegraf, Grafana) installed and configured.

8.2. Exit Criteria

- All test scripts completed successfully
- No critical problems encountered
- All non-critical problems are logged
- All test logs are captured
- All post-test notifications sent

8.3. Suspension Criteria

- Not all test scripts will complete
- Critical problems are encountered and logged
- Hardware errors prevent the completion of the test

2. *Schedule*

Test milestones and item transmittal events:

#	Stage	Period	Notes
1.	Test strategy design		
2.	Test plan creation		
3.	Draft of NFR definition		
4.	Setting up test environment		
5.	Setting up load generator tool and tools for collection and visualization of metrics		
6.	Test data preparation		
7.	Script/Scenarios design		
8.	Automation test running from CI/CD		
9.	Tests round for a system whose data source is file system		
10.	- Perform capacity testing		Performing a smoke test before each run
11.	- Perform scalability testing		Performing a smoke test before each run
12.	- Perform volume testing		Performing a smoke test before each run
13.	- Perform long-time testing		Performing a smoke test before each run
14.	- Perform regular load and stress testing		Performing a smoke test before each run
15.	Test results analysis		
16.	Test results reporting		
17.	Switch data source from file system to DB		
18.	Configuring system settings and permissions for roles after data source switchover		
19.	Updating scripts/scenarios		
20.	Tests round for a system whose data source is file system		
21.	- Perform capacity testing		Performing a smoke test before each run
22.	- Perform scalability testing		Performing a smoke test before each run
23.	- Perform volume testing		Performing a smoke test before each run
24.	- Perform long-time testing		Performing a smoke test before each run
25.	- Perform regular load and stress testing		Performing a smoke test before each run
26.	Test results analysis		
27.	Test results reporting		
28.	Prepare complex report on entire performance testing process		

3. Risks and contingencies

Risks:

- A significant difference in configuration from the production environment
- Performance testing results can be essentially different even in case of minor difference in think times, arrival rate and test duration
- During the execution of the tests, some major performance or functional problems that may require code changes, creation of a new build may be discovered and in that case it may be necessary to repeat the load test from the beginning
- Load test should be performed against a build that is solid enough, and that has been functionally tested, after code is complete. Failure to follow this rule may result on rework to update test scripts for every new build, plus the load test may need to be repeated from the beginning. This will affect the schedule
- Performance testing tool is not capable of identically reproducing real life scenarios - so results could only be trusted as having limited reliability level
- Delays in setting up a test environment or load generator
- Possible difficulties with setting up system parameters after switching from the file system to DB in terms of bringing it to a state exactly corresponding to what it was before the switch.