**Report about conducted scalability test**

**Date:** 1-3 August 2022

**Author:** Oleksandr Maksymenko

**Application:** BlogEngine.NET version 3.2

**Environment:** Host machine where script was run + Virtual Machine where blog is deployed

**Environment configuration (RAM, CPU etc.):**

|  |  |
| --- | --- |
| **RAM** | 4096 MB |
| **CPU** | 1 Core CPU (Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz) |
| **System Type** | Windows 10 64-bit |

1. **Why such testing was conducted:** To find threshold for degradation depending on the quantity of the posts.
2. **Test script description:** Current script is combined script of 3 different role user scenarios: Anonymous, Admin and Editor.
   1. **Anonymous Script:** contains different user scenarios which are chosen with different probabilities (Open Home Page, Open Random Date, Open Predefined Date, Search Post by Name, Open large calendar, Open contacts). First three scenarios are opening random page with posts (in 50% cases). All scenarios (except Open Contacts) have Open Page script, which is used randomly in ~80% cases. Detailed description can be found on screenshots and in tables below.
   2. **Admin Script:** contains actions for Admin flow. Firstly “user” opens Home page, Logs In, after that starts loop which executes 10 times: Open Admin Page -> Open Users Page -> depending on current number of users -> Create User OR Delete User. After loop ends “user” will be logged off.
   3. **Editor Script:** contains actions for Editor flow. Firstly “user” opens Home page, Logs In, after that starts loop which executes 50 times: Open Predefined Date -> Open Random Page (in case editable post was not found on first page) -> Open Random Post -> Edit Post. After loop ends “user” will be logged off.

**Diagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generated**

Diagram

Description automatically generatedDiagram

Description automatically generated

# Tests preconditions and configurations

**Test run preconditions:**

* 100, 1000, 2000 or 5000 pregenerated posts on predefined dates
* 4 pictures loaded to Blog
* 1000 pregenerated media posts (with one 1Mb picture)
* CSV file with 10 predefined dates
* CSV file with Admin credentials
* CSV file with Editor credentials

**Monitored Key Performance Indicators (KPI):**

* Throughput (transactions per second)
* Error rate, %
* Average response time, ms
* Median response time, ms
* 90% response time, ms
* 95% response time, ms

**Why that Load Model was chosen:** task 7 test execution showed saturation point on ~90 users, so number of anonymous users was set to 60 users, editor users to 8, and admin users remained the same at 2. Overall number of users equals to 70 in that load test, which is close to saturation point, but at the same time system does not work on 100% CPU load all the time (from Detailed test results we can see average CPU load was 76% for that config), which means we should have stable processing time during execution. Rampup period was set to 2 minutes, and other 8 minutes we had stable load.

**Load Model:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Users** | **Ramp up (s)** | **Duration (s)** |
| **Anonymous** Users | 60 | 120 | 600 |
| **Admin** Users | 2 |
| **Editor** Users | 8 |

**Comparison of Anonymous script step probabilities:**

|  |  |
| --- | --- |
| **Action** | **Probability** |
| Open Home Page | 15% |
| Open Random Date | 10% |
| Open Predefined Date | 30% |
| Search by Name | 30% |
| Open Large Calendar | 10% |
| Open Contacts | 5% |

|  |  |
| --- | --- |
| **Action:** Open Random Page | **Probability** |
| Yes | 50% |
| No | 50% |

|  |  |
| --- | --- |
| **Action:** Open Post | **Probability** |
| Yes | 80% |
| No | 20% |

|  |  |
| --- | --- |
| **Action:** Open First or Random Post | **Probability** |
| First | 35% |
| Random | 65% |

|  |  |
| --- | --- |
| **Action:** Send Comment | **Probability** |
| Yes | 20% |
| No | 80% |

# Short summary on conducted tests:

**Test with different number of the text posts:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Posts** | **100** | **1000** | **2000** | **5000** |
| **Throughput (transactions per sec)** | 22.09 | 22.11 | 21.05 | 15.81 |
| **Error rate, %** | 0 | 0.01 | 0.68 | 1.29 |
| **Average response time, ms** | 182.26 | 170.76 | 345.43 | 1413.97 |
| **Median response time, ms** | 126.5 | 122.5 | 621.5 | 3010.5 |
| **90%, ms** | 1210 | 1159 | 1955.6 | 5070.5 |
| **95%, ms** | 1704.35 | 1913.4 | 2578.95 | 5747.65 |
| **Average CPU load, %** | 55 | 61 | 78 | 90 |

In that table we can see that all metrics have similar data for 100 and 1000 posts, while for 2000 all KPI increased: ~2x for AVG response time, ~4x for Median, +65% for 90 percentile and +43% for 95 percentile. Average CPU load also increased from 55-60% up to 78% and on Detailed Test results screenshots we can see higher amount of 100% load peaks for 2000 posts run compared to 100 and 1000 posts. When we include 5000 text posts test run to comparison – difference becomes more impactful, as all KPI has grown again, including Median response time ~3 seconds, and average CPU load ~90%. Error rate also was growing for each run with higher number of posts.

**Test with mixed posts (text + media):**

|  |  |  |
| --- | --- | --- |
| **Number of Posts** | **2000 text posts** | **1000 text posts + 1000 media posts** |
| **Throughput (transactions per sec)** | 21.05 | 21.44 |
| **Error rate, %** | 0.68 | 0.97 |
| **Average response time, ms** | 345.43 | 270.26 |
| **Median response time, ms** | 621.5 | 223.5 |
| **90%, ms** | 1955.6 | 1396.5 |
| **95%, ms** | 2578.95 | 1824.3 |
| **Average CPU load, %** | 78 | 70 |

Here we compare KPI for test runs with 2000 text only posts and 2000 mixed posts (1000 text and 1000 with 1Mb picture). Throughput was similar in both test runs, error rate was higher for mixed posts case, but all other metrics showed better results for mixed posts comparing with pure text posts: average response time was 22% lower, median lost 64%, both 90 and 95 percentiles were 30% lower than in pure text posts. Average CPU load rate also decreased from 78 to 70 percent.

# Detailed test results:

## Text posts only

### 100 text posts

**Composite Graph (Active Threads, Throughput, Average Response Time, Errors per sec)**

Chart, histogram

Description automatically generated

**Aggregate Report**

Calendar

Description automatically generated with low confidence

**All transaction response times**

Chart, histogram

Description automatically generated

**CPU load**

Chart

Description automatically generated with medium confidence

**Memory usage**

A screenshot of a video game

Description automatically generated

**Garbage Collector**

A screenshot of a computer

Description automatically generated with medium confidence

**Cache**

A screenshot of a computer

Description automatically generated with medium confidence

### 1000 text posts

**Composite Graph (Active Threads, Throughput, Average Response Time, Errors per sec)**

Chart

Description automatically generated

**Aggregate Report**

A picture containing text, screen

Description automatically generated

**All transaction response times**

Chart, histogram

Description automatically generated

Response time spike at 19:27:35 happened at the same moment when CPU load was down to ~40% instead of average 60%.

**CPU load**

A screen shot of a video game

Description automatically generated with medium confidence

**Memory usage**

A screenshot of a video game

Description automatically generated

**Garbage Collector**

A screenshot of a computer

Description automatically generated with medium confidence

**Cache**

A screenshot of a computer

Description automatically generated with medium confidence

### 2000 text posts

**Composite Graph (Active Threads, Throughput, Average Response Time, Errors per sec)**

Graphical user interface, chart

Description automatically generated

**Aggregate Report**

A picture containing text, computer

Description automatically generated

**All transaction response times**

Chart, histogram

Description automatically generated

Response time had its first peak at 19:49-19:51:30, and second peak (at 19:55:15) happened after CPU reached 100% and after that it was 80-100%.

**CPU load**

A picture containing text, night, dark

Description automatically generated

**Memory usage**

A screenshot of a computer

Description automatically generated with medium confidence

**Garbage Collector**

A screenshot of a computer

Description automatically generated with medium confidence

**Cache**

A screenshot of a computer

Description automatically generated with medium confidence

No application restarts, only w3wp and w3wp#1 switch

### 5000 text posts

**Composite Graph (Active Threads, Throughput, Average Response Time, Errors per sec)**

Chart, histogram

Description automatically generated

Response time increased when CPU load reached 100%, and despite it was down shortly after that – response time did not go down together with it, as average CPU load during execution was ~90%.

**Aggregate Report**

A picture containing text, outdoor, computer

Description automatically generated

**All transaction response times**

Chart

Description automatically generated

**Errors**

Graphical user interface

Description automatically generated

**CPU load**

A screenshot of a video game

Description automatically generated

**Memory usage**

A screenshot of a video game

Description automatically generated

**Garbage Collector**

A screenshot of a computer

Description automatically generated with medium confidence

**Cache**

A screenshot of a computer

Description automatically generated with medium confidence

No application restarts, only w3wp and w3wp#1 switch.

## Mixed posts

### 1000 text + 1000 media posts

**Composite Graph (Active Threads, Throughput, Average Response Time, Errors per sec)**

Chart

Description automatically generated

**Aggregate Report**

A picture containing text, electronics, computer

Description automatically generated

**All transaction response times**

Chart, histogram

Description automatically generated

Response time spike at 17:35:35 happened at the same moment when CPU load was down to ~23% instead of average 70%.

**CPU load**

Background pattern

Description automatically generated

**Memory usage**

A screenshot of a video game

Description automatically generated

**Garbage Collector**

Graphical user interface

Description automatically generated

**Cache**

A screenshot of a computer

Description automatically generated with medium confidence

# Conclusion:

* We executed 5 tests with different number of posts and with different content in them.
* In first part we executed test script for 100, 1000, 2000 and 5000 pure text posts. Comparison of their KPIs can say that BlogEngine application behaves similar for 100 and 1000 posts but somewhere between 1000 and 2000 posts application becomes to degrade, as we can see slightly worse results for 2000 posts case, and much worse situation for 5000 posts case. In Conclusion we can say that **for current environment configuration** (1 Core CPU and 4 GB RAM) **number of posts without degradation** when users are not complaining on slow application **is less than 2000 pure text posts**.
* In second part we came back to 1000 text posts and created 1000 media posts with 1Mb picture. When we compare KPIs of 2000 pure text posts and 2000 mixed posts we can slightly lower response time KPIs for mixed posts case despite of potentially higher time to render pictures on the page. It could happen because of application cache, which Entries were practically not growing during test run of mixed posts and Cache Output Hits graph was constantly growing. To find degradation threshold for mixed posts cases more tests with different number of posts are required, including cases with different ratio of text and media posts.