**Prerequisites:**

* Generated 100 posts
* Generated 1000 posts

**Description:**

Implement Anonymous user scenario (see Main script, Anonymous script, Open post script algorithms) as in Task 3.

**Update the script/scenario to implement probabilities usage:**

1. Home Page: 15%

2. Open Random Date: 10%

3. Open Predefined Date: 30%

4. Search by Name: 30%

5. Open Large Calendar: 10%

6. Open Contacts: 5%

Open Random page (yes/no):

50% / 50%

Open post (yes/no):

80% / 20%

Random or First (yes/no):

65% / 35%

Comment (yes/no):

20% / 80%

* Perform Smoke testing.
* Perform Capacity testing for the test from Task 3 (the same probabilities for all branches) with two sets of generated posts (100 and 1000).
* Perform Capacity testing for Task 6 with two sets of generated posts (100 and 1000).
* Compare results between:
  + Task 3 and Task 6: 100 posts
  + Task 3 and Task 6: 1000 posts
  + Task 6: 100 and 1000 posts
* Perform analysis of the results with summary and detailed explanation of changes in the behavior.
* Document results.

During the testing gather main business, application and system metrics and include them in the comparison.

**User role**: anonymous

**Note**: the task is to be done without third party controllers.

**Goals:**

* Get an experience of smooth update of the script according to new requirements and cases
* Get an experience in regression testing (like after new version uploaded)
* Get an experience in volume testing
* Improve skills in comparison and analysis

**Results:**

For measuring capacity in all cases was selected such workload model:

Number of users = 300

Rampup = 900 sec

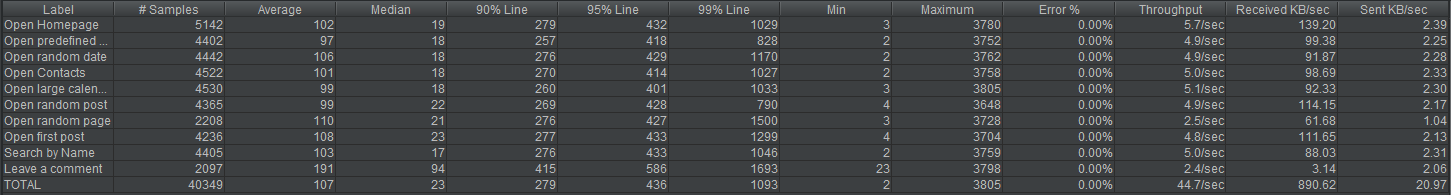
Duration = 900 sec

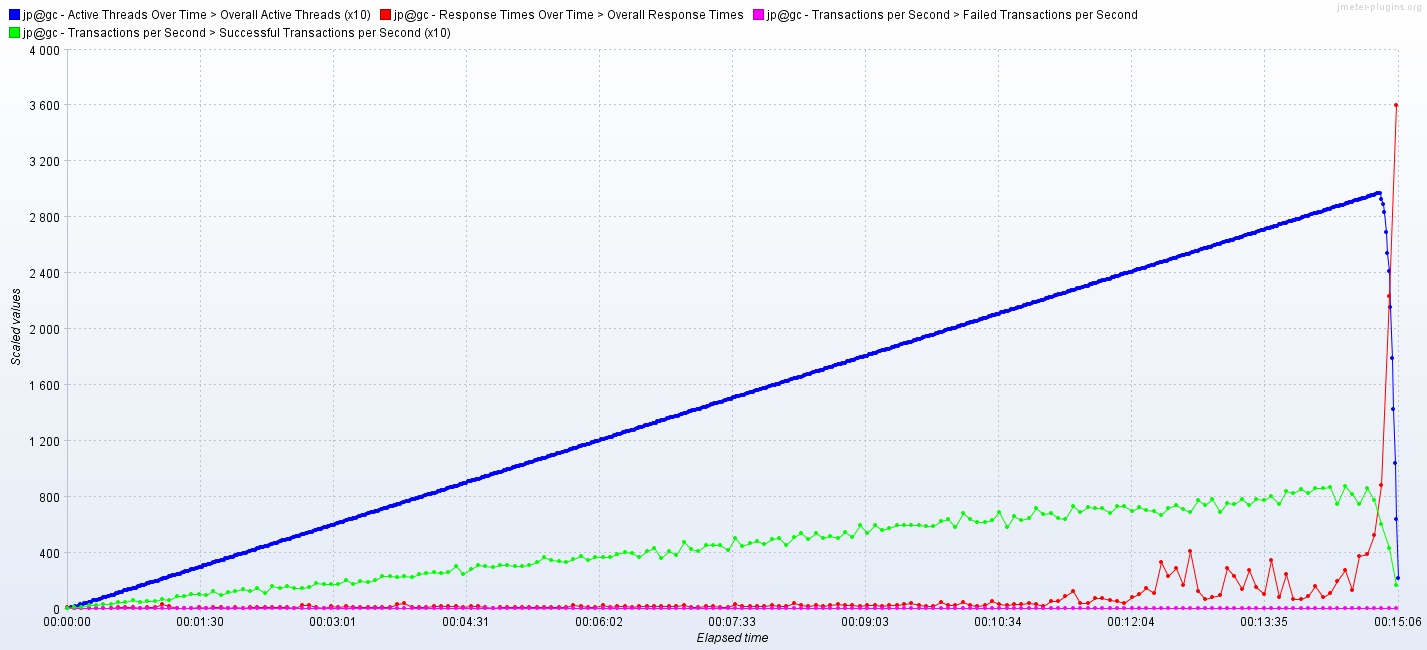
Workload profile is different for both of scripts and the main dependency is frequency of actions that should be performed by users, what is mentioned above. Also both of scripts were run twice for different amount of posts: 100 and 1000. Detailed info run by run and comparison described below.

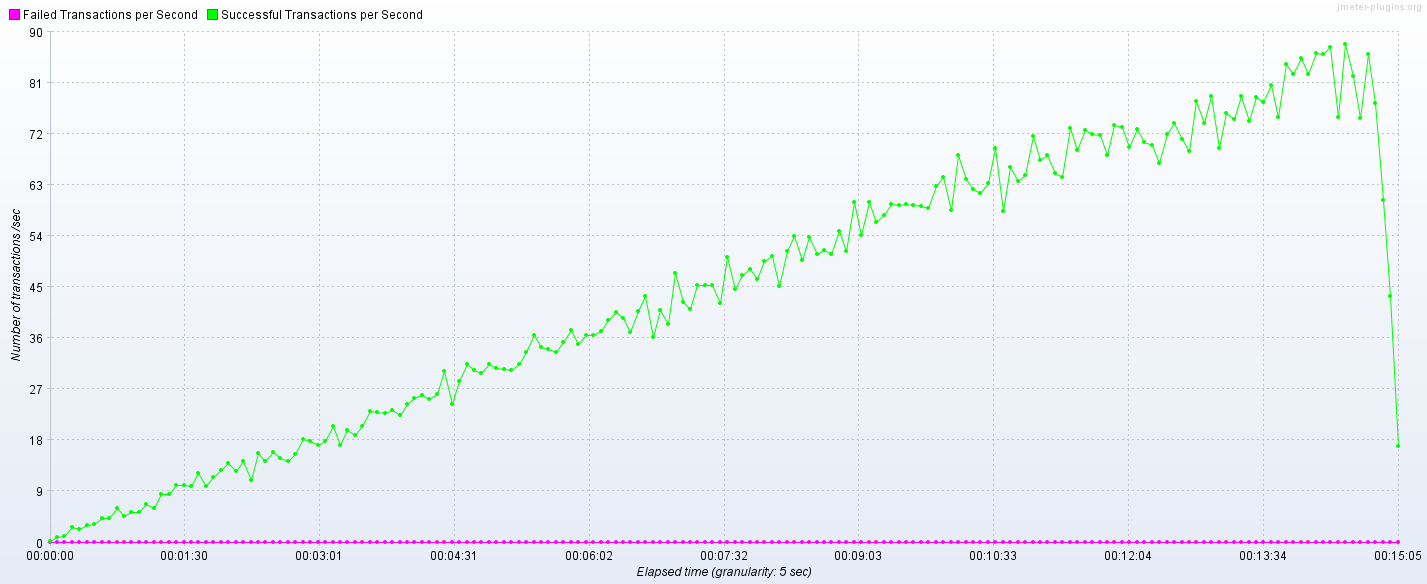
* 1. **Task 3: 100 posts**

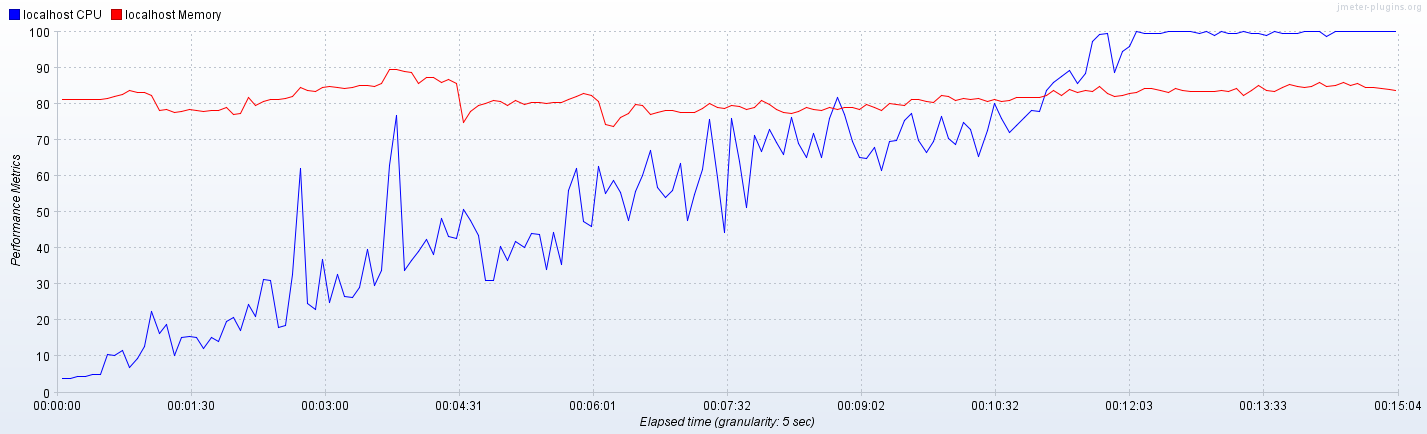
In this script frequency of each action is the same. Amount of data to handle is 100 earlier generated posts.

From the graphics we can see, that till 11:04 system is in comfort zone, but then response time starts growing. CPU also don`t fall from this time, so in approximately 11:04 capacity point was reached. At this time 220 users are using the application, so it`s approximate maximum amount that system can handle without problem with such amount of data in 100 posts. Throughput at this time is 72 transactions per second.







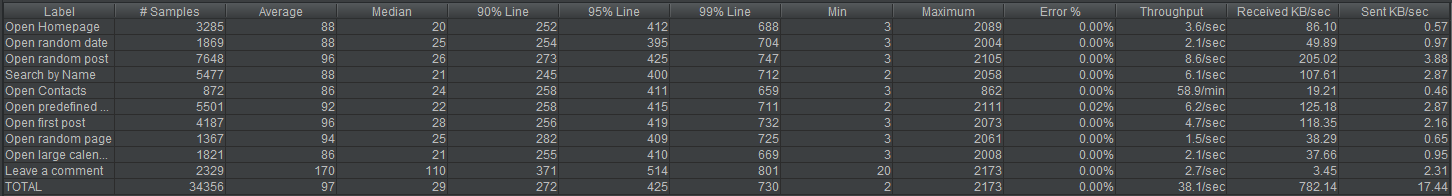


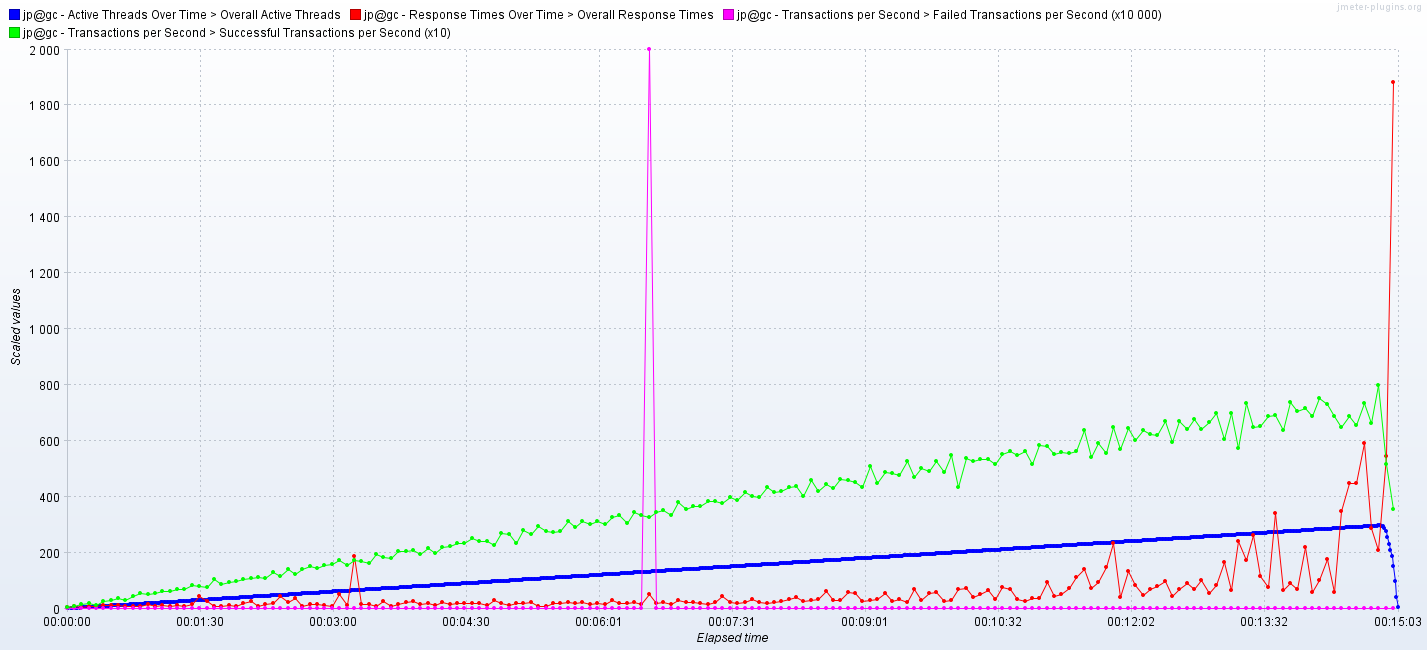
* 1. **Task 6:100 posts**

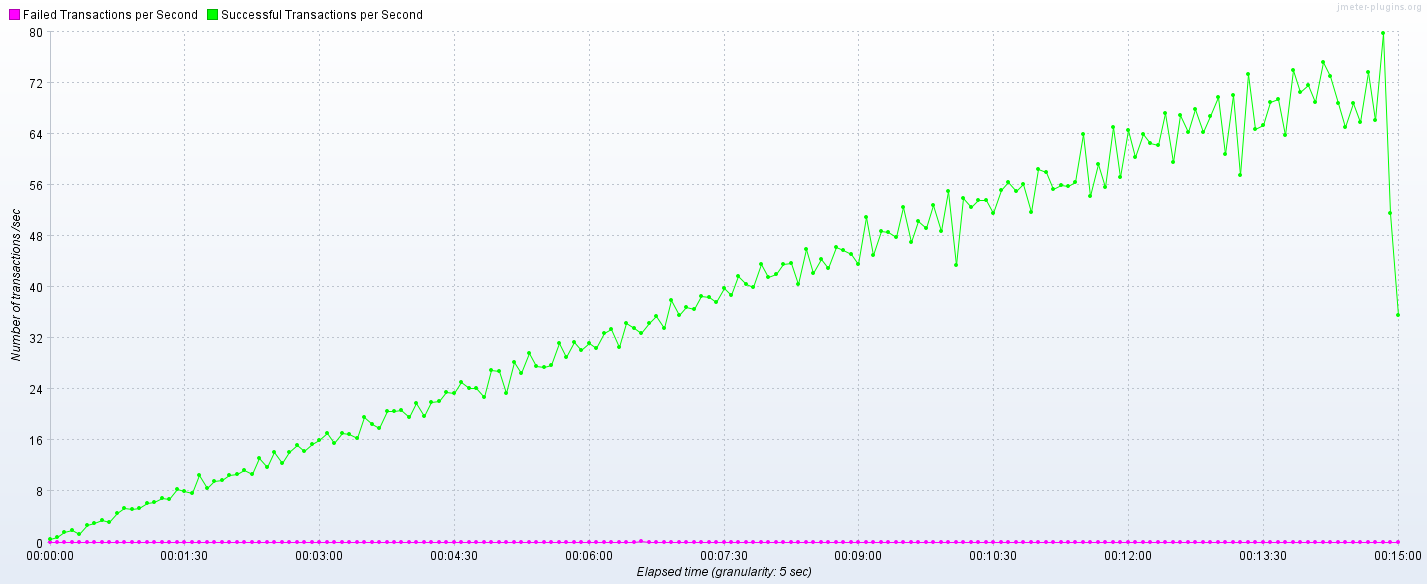
In this script frequency of each action was changed. So, frequency of requests, that led to performing more actions was raised. For example percentage of opening predefined date and performing search is greater in comparison with other possible random actions. Such request as Open Contacts is executing less times, because it is like endpoint. Also frequency of opening post was raised till 80 %.

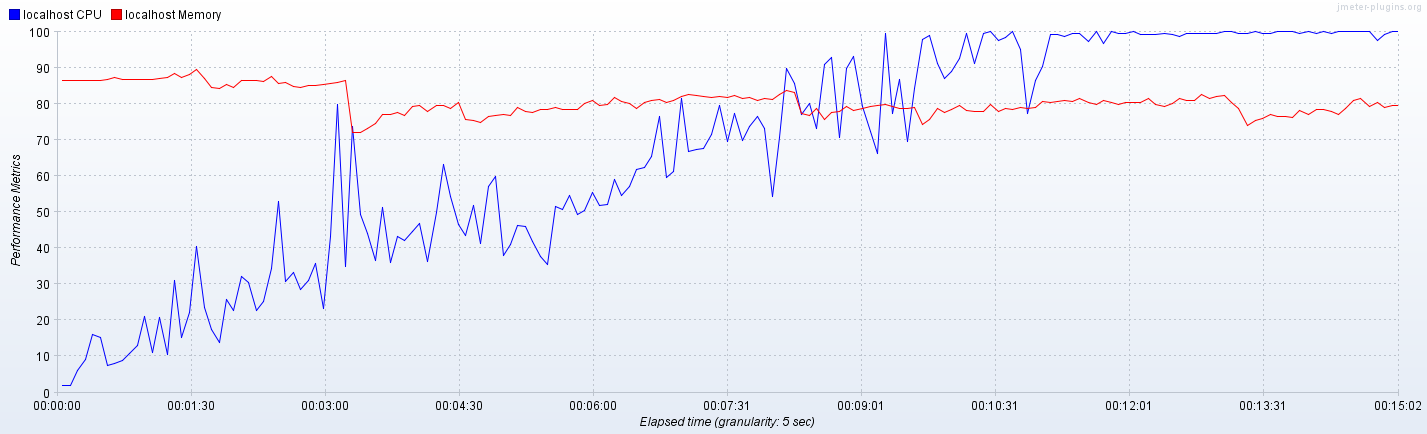
Amount of data to handle is 100 earlier generated posts. From the graphics we can see, that till 9:05 system is in comfort zone, but then response time starts growing, throughput continue growing but not so fast and with more rapid changes, CPU firstly reached 100 %.

We may say, that in approximately 9:05 from the start capacity point was reached. At this time 180 users are using the application, so it`s approximate maximum amount that system can handle without problem with such amount of data in 100 posts. Throughput at this time is 52 transactions per second.





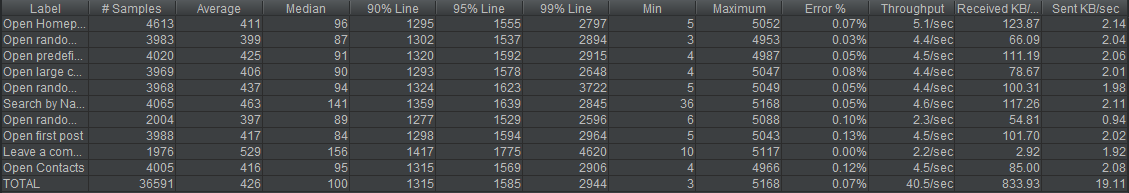


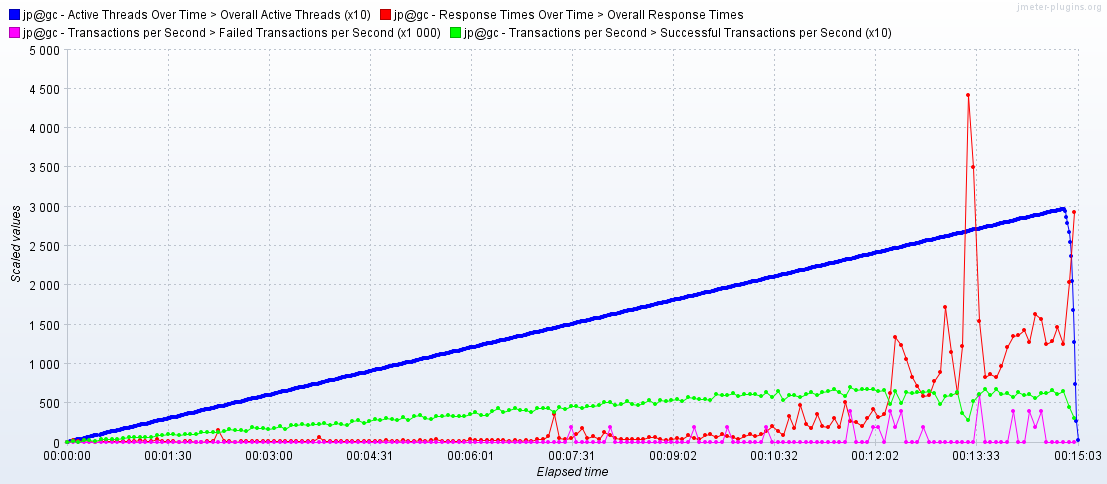


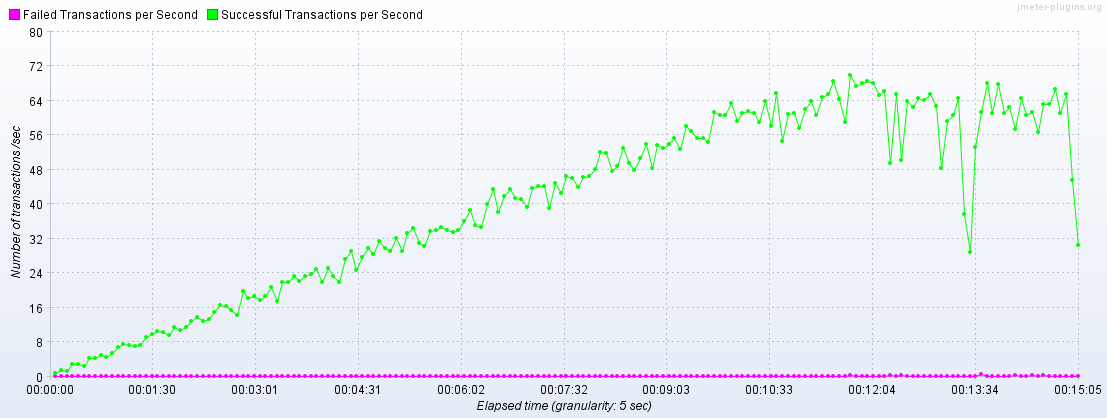
* 1. **Task 3: 1000 posts**

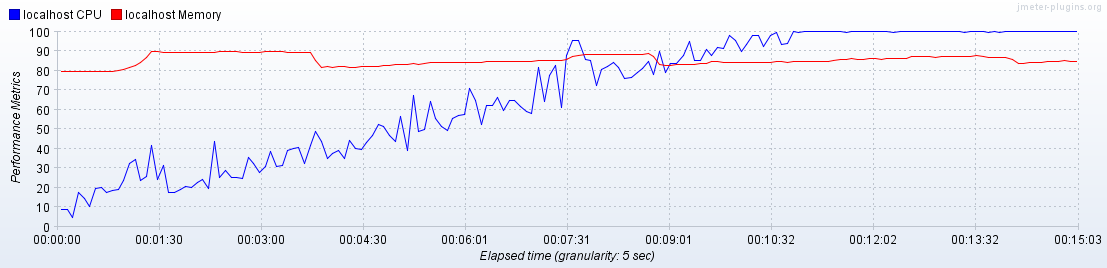
In this script frequency of each action is equal. Amount of data to handle was increased to 1000 earlier generated posts.

From the graphics we can see, that till 10:05 system is in comfort zone, but then response time starts growing and throughput not a lot. CPU also don`t fall a lot from this time till reaching 100 %, so in approximately 10:05 capacity point was reached. At this time 205 users are using the application, so it`s approximate maximum amount that system can handle without problem with amount of data in 1000 posts and equal frequency of executing different actions. Throughput at this time is 64 transactions per second.







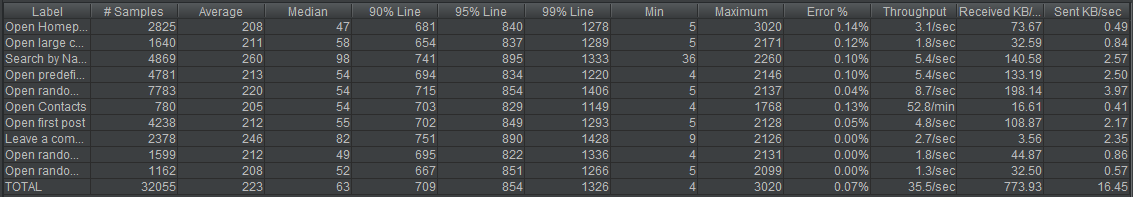


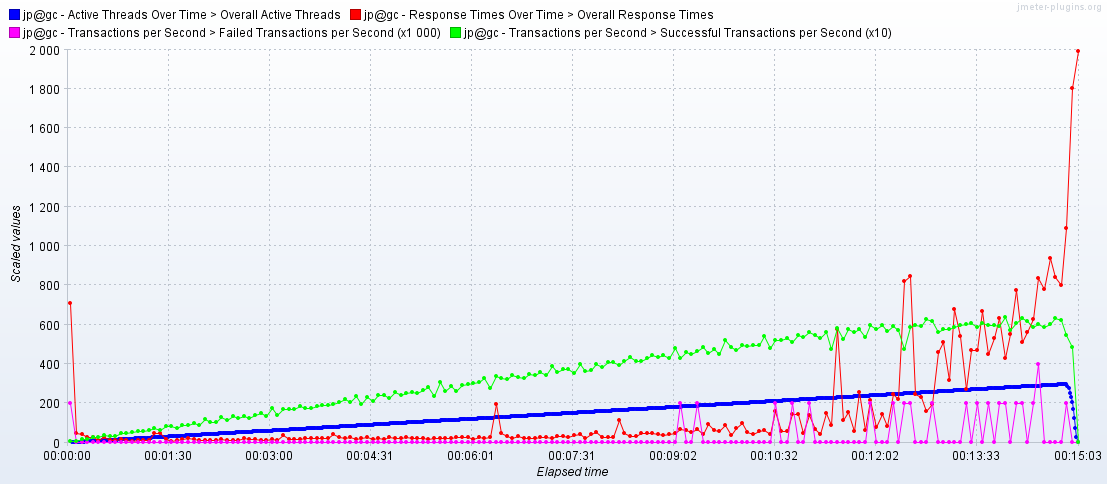
* 1. **Task 6: 1000 posts**

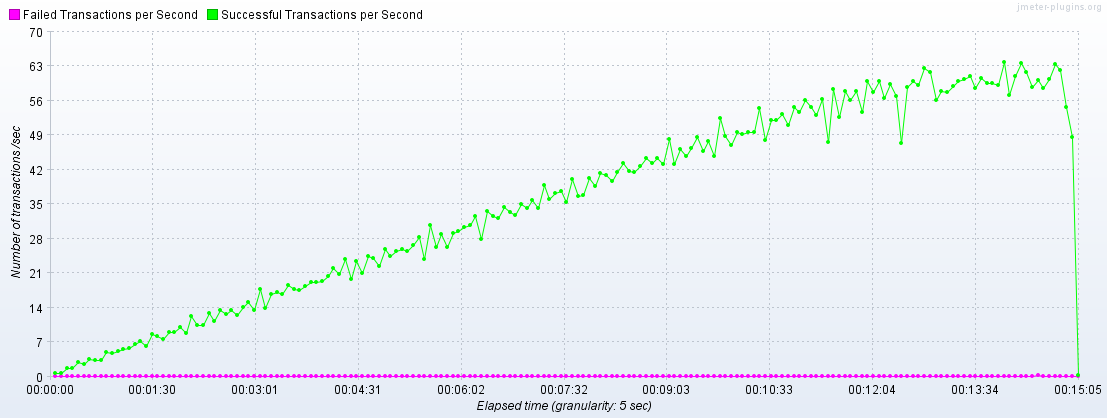
As was already mentioned frequency of each action in this script was changed.

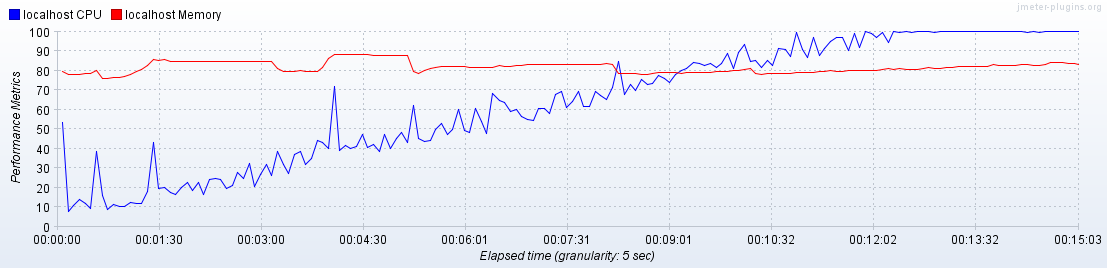
Amount of data to handle is 1000 earlier generated posts. From the graphics we can see, that till 08:20 system is stable, but then response time starts growing, throughput continue growing but not so fast and with more rapid changes, CPU firstly reached 100 %.

We may say, that in approximately 08:20 from the start capacity point was reached. At this time 165 users are using the application, so it`s approximate maximum amount that system can handle without problem with such amount of data in 1000 posts. Throughput at this time is 44 transactions per second.









**Summary**

From analyzing all of the runs separately now we may compare them:

**1) Task 3 and Task 6: 100 posts**

Saturation point:

11:04 vs 9:05

Throughput:

72 vs 52

Amount of users:

220 vs 180

So, such results are expected, because we changed frequency of actions, forcing the system to perform **more** **amount** of actions from the one side and to perform more **resource intensive** requests from the other side, such as Open Predefined date, Search by Name and Open post.

**2) Task 3 and Task 6: 1000 posts**

Saturation point:

10:05 vs 8:20

Throughput:

64 vs 44

Amount of users:

205 vs 165

For such comparison situation is the similar, the only thing that was changes is amount of posts. That`s why such results are different from the previous, but global difference can be explained from the next pair.

**3) Task 6: 100 and 1000 posts**

Saturation point:

09:05 vs 8:20

Throughput:

52 vs 44

Amount of users:

180 vs 165

Here we can see, how performance can be changed if increase amount of data to operate with. So, due to this kind of volume testing we can understand, that by increasing the volume of data we get our performance lower. Also last run of 6th script with 1000 posts have the lowest values metrics from four of them, because it combines both of cases, as huge amount of data and more amount of requests.