Working on the \*\*\*\* stabilization task, we reworked and improved existing code. While we’re still working on this task, we have some preliminary results.

This report represents performance tests results after we reworked incoming request handling (async IO, concurrent request execution, etc.). The tests were made using JMeter.

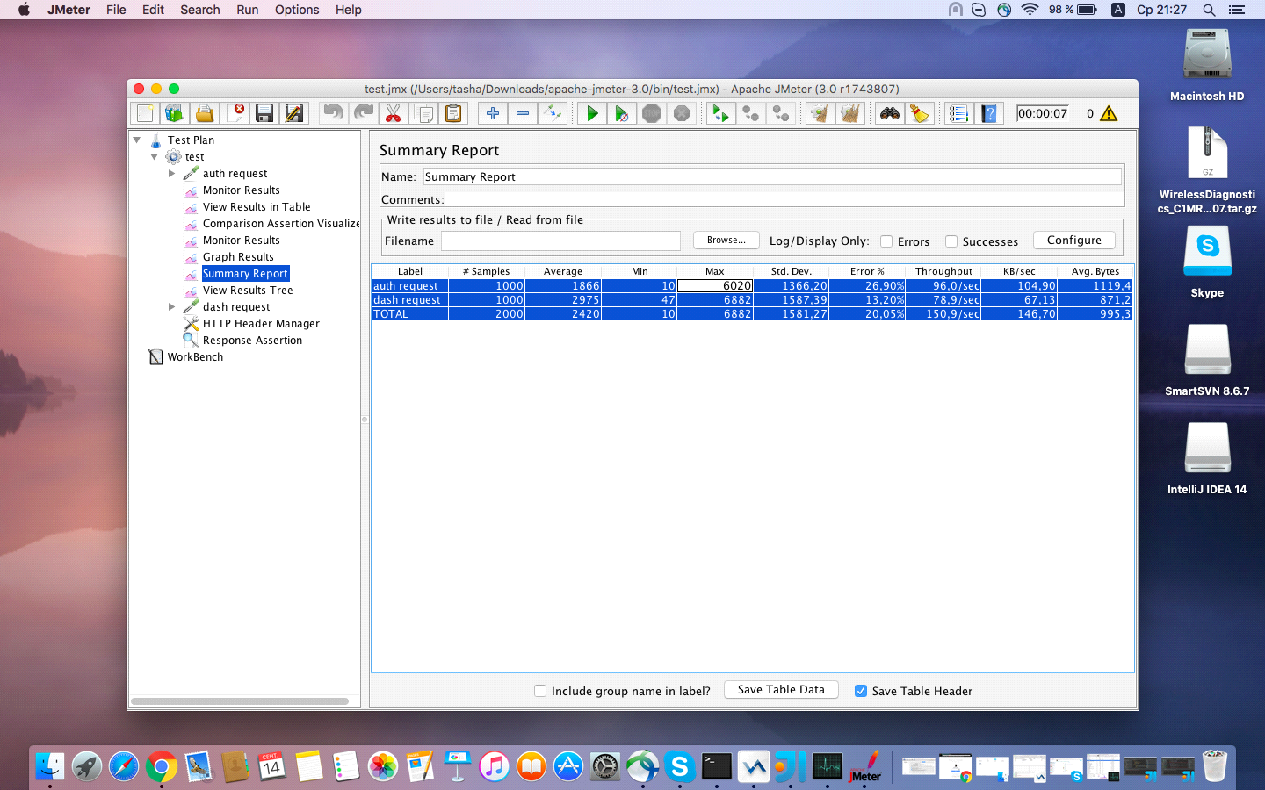
We have simulated and tested **2** main cases, both of them we have applied to the **old** and **new \*\*\*\*** implementations, for comparison reason.

**Test case #1**

* We have created an *HTTP Request* sampler with *Thread Group*, which represents a page **authentication** request that each thread (user) will access**: 1000 threads over 3-4 seconds;**
* Use parsed **sessionID** from first request for the next direct request to the some specific servlet(as example **dashboard);**

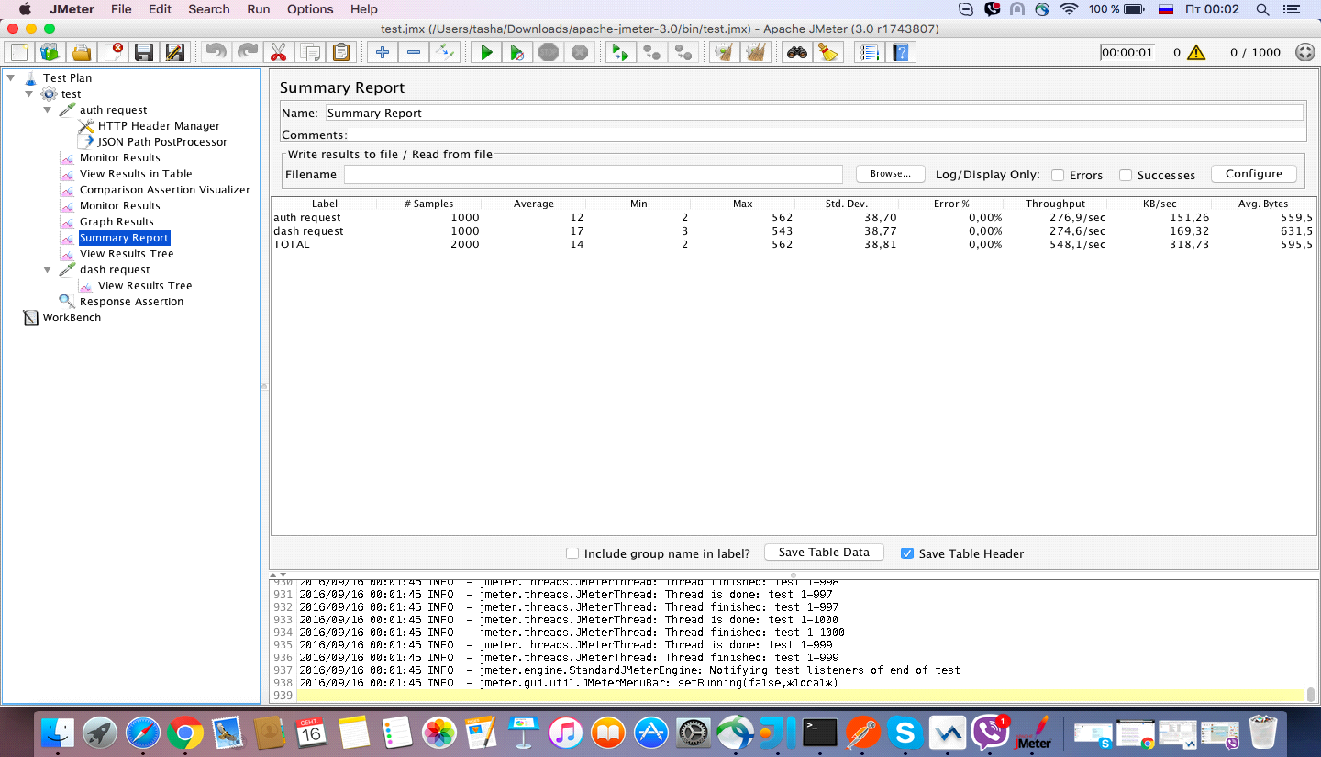
**Results**

**Old implementation:**



**Pic 1.1 – Summary report (case #1) – old implementation**

**New implementation:**

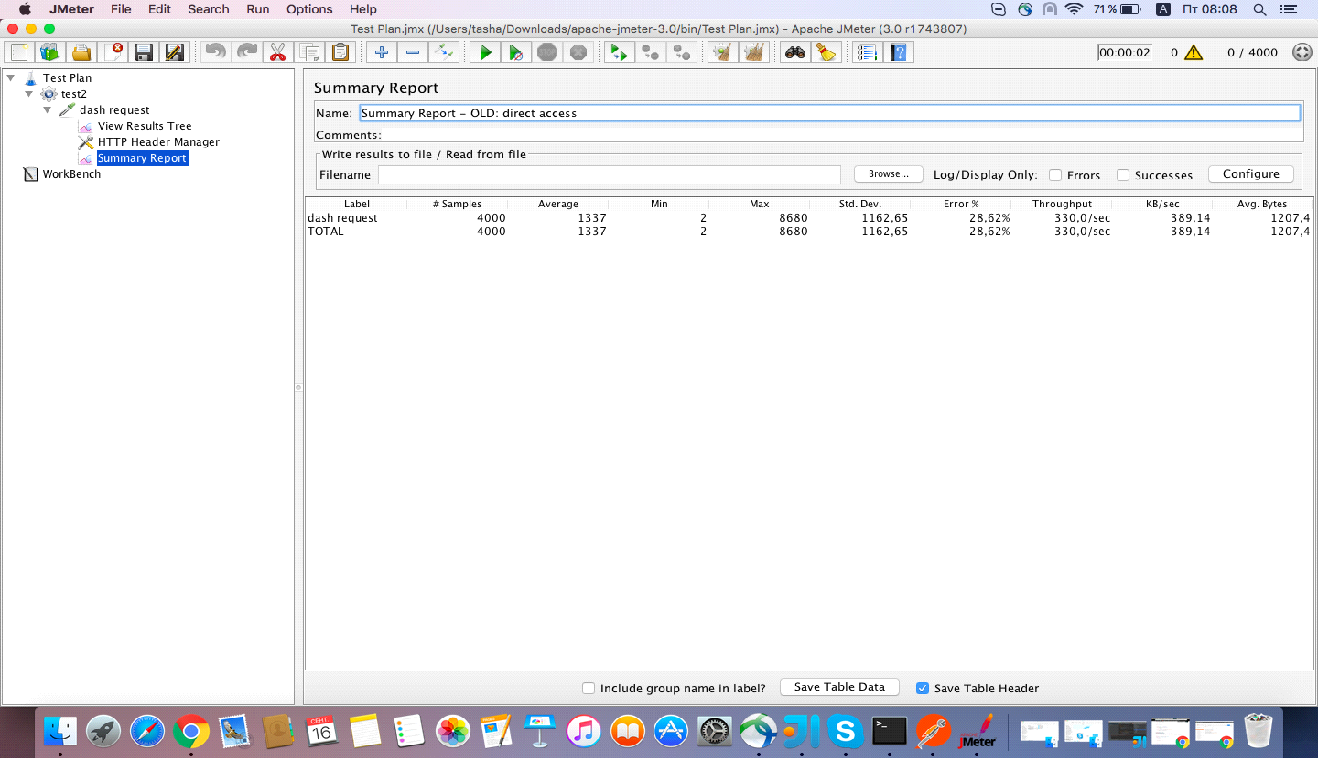


**Pic 1.2 – Summary report (case #1) – new implementation**

**Test case #2**

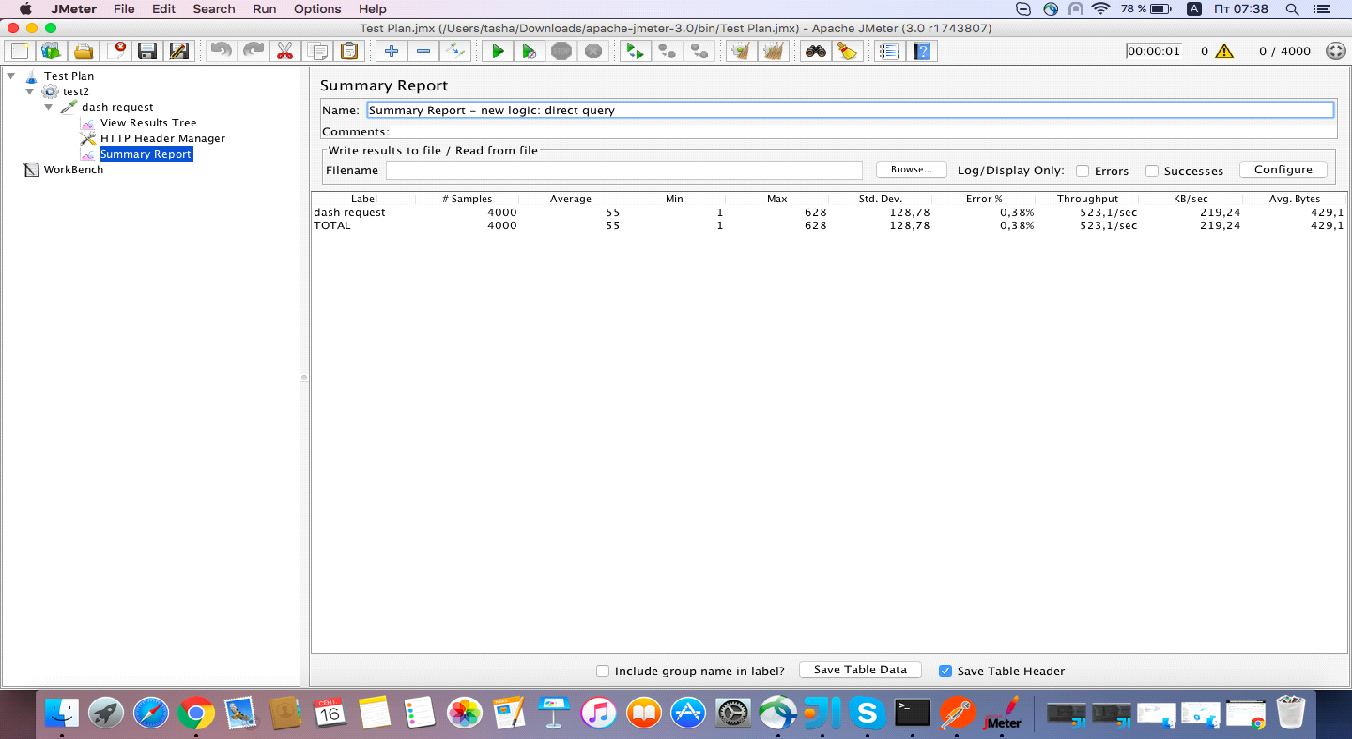
* Simulate the direct request to the some specific servlet without authentication: : **4000 threads over 3-4 seconds;**

**Old implementation:**



**Pic 1.3 – Summary report (case #2) – old implementation**

**New implementation:**



**Pic 1.4 – Summary report (case #2) – new implementation**

**Interpretation**

The summary report shows values on the measurement JMeter has done while calling the test page as if many users are calling the page.

Below you can see the details and explanations regarding these parameters and numbers.

**Samples:**Samples denote to the number of http request ran for given thread. Like we have one http request and we run it by 5 users, then the number of samples will be 5x1=5.  
Same if the sample ran two times for the single user, then the number of samples for 5 users is 5x2=10.

1 test case Old -> 1000 requests; New -> 1000 requests  
 2 test case Old -> 4000 requests; New -> 4000 requests

**Average:**  Average is the average response time for that particular http request. The response time is represented in millisecond. On the corresponding picture you can see that for the first label the number of sample is 4, because that sample runs 2 times for a single user, and we ran the test with 2 users. So, for 4 samples the average response time is 401 ms.

(less is better)

1 test case Old -> 2420 milliseconds; New -> 14 milliseconds  
 2 test case Old -> 1377 milliseconds; New -> 55 milliseconds;

**Min:**Min denotes to the minimum response time taken by the http request. Like for the above image the minimum response time for first four samples is 266 ms. It means one http request responded in 266 ms out of four samples.

(less is better)  
1 test case Old -> 10 milliseconds; New -> 2 milliseconds  
 2 test case Old -> 2 milliseconds; New -> 1 milliseconds

**Max:**Max denotes to the maximum response time taken by the http request. Like for the above image the maximum response time for first four samples is 562 ms. It means one http request responded in 562 ms out of four samples.

(less is better)  
1 test case Old -> 6882 ms; New -> 562 ms  
 2 test case Old -> 8680 ms; New -> 628 ms

**Std.Deviation:** This shows how many exceptional cases were found which were deviating from the average value of the receiving time.   The lesser this value more consistent the time pattern is assumed.

(less is better)  
1 test case Old -> 1581.27 ms; New -> 38.81ms   
2 test case Old -> 1162.65 ms; New -> 128.78 ms  
  
**Error %:** This denotes the error percentage in samples during run. This error can be of 404(file not found), or may be exception or any kind of error during test run will be shown in Error %. In the above image the error % is zero, because all the requests ran successfully.

(less is better)

1 test case Old -> 20.05 – 30%; New -> 0-1%  
2 test case Old -> 28.6-40%; New -> 0-1%