**Req. 7.4** – An actor who is not authenticated must be able to search for a position using a single key word that must be contained in its title, its description, its profile, its skills its technologies or the name of the corresponding company.

Technical details of the computer on which the test has been executed:

* Ram: 8,192 (1x) MB, DDR3L RAM (1,600 MHz)
* CPU: Intel Core i7-5500U
* Disco duro: 1 TB HDD - 5,400 rpm
* Tarjeta de red: Gigabit Ethernet LAN - 10BASE-T/100BASE-TX/1000BASE-T

**Test case description**:

* The user goes to search view.
* Searches for positions using the key word ‘r’.

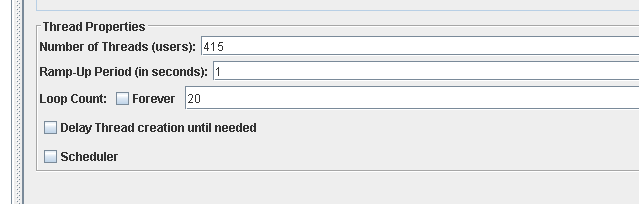
**Maximum workload test case.** 415 concurrent users and 20 of loop count:

Figure 1. Configuration

This is the maximum workload of the test case without any crash or excesive delay. As we can see in the picture below, we don’t have any errors and the average time per request is pretty acceptable.

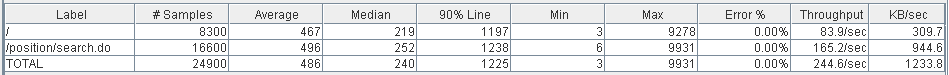


Figure 2. Aggregate report

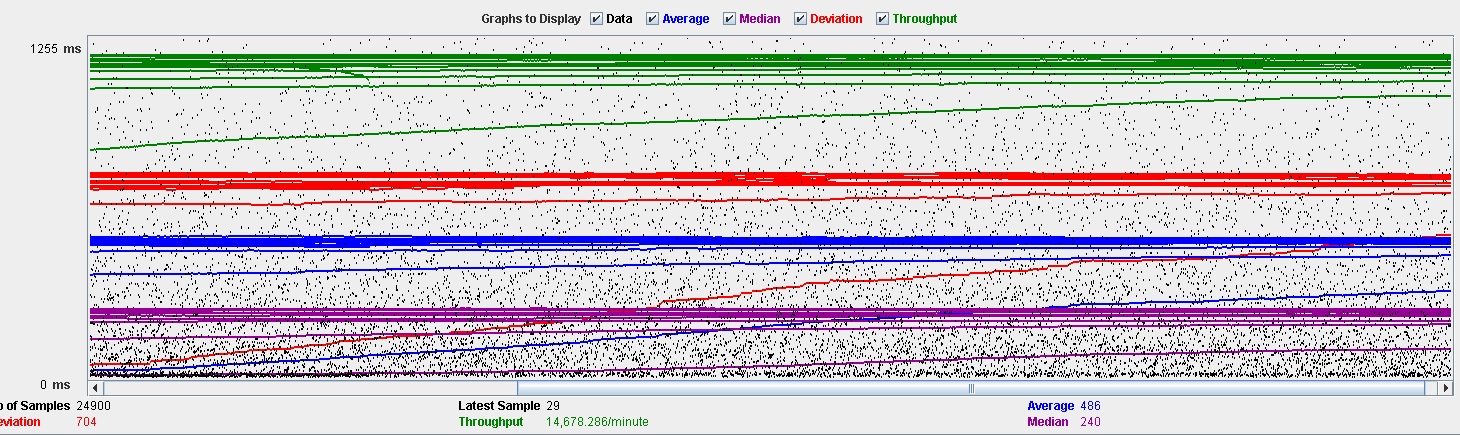


Figure 3. Stabilized output graph report

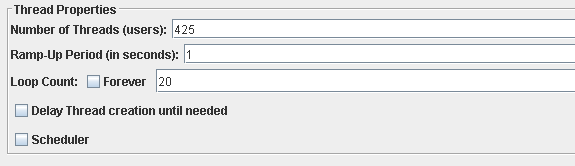
 **Overload test case:** 425 concurrent users and 20 of loop count:

Figure 4. Configuration

Even though the average time per request is still acceptable, it begins to produce some errors as we can see in the following picture. The errors are always the same: I/O exception (java.net.SocketException) caught when processing request: Connection reset by peer: socket write error

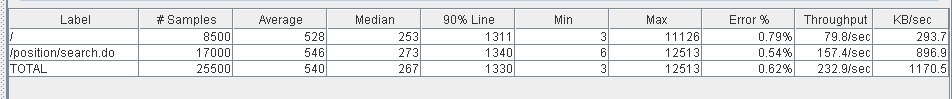
This exception is not related with the implementation of our application, but with tomcat. The system can’t handle this number of concurrent users properly.

Figure 5. Aggregate report

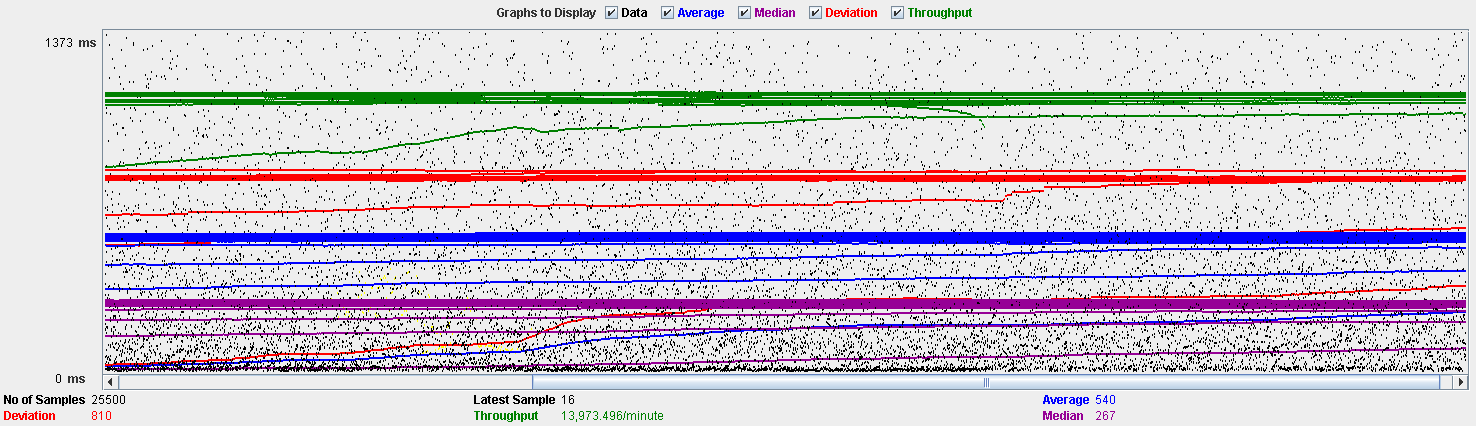
****

Figure 6. Stabilized output graph report

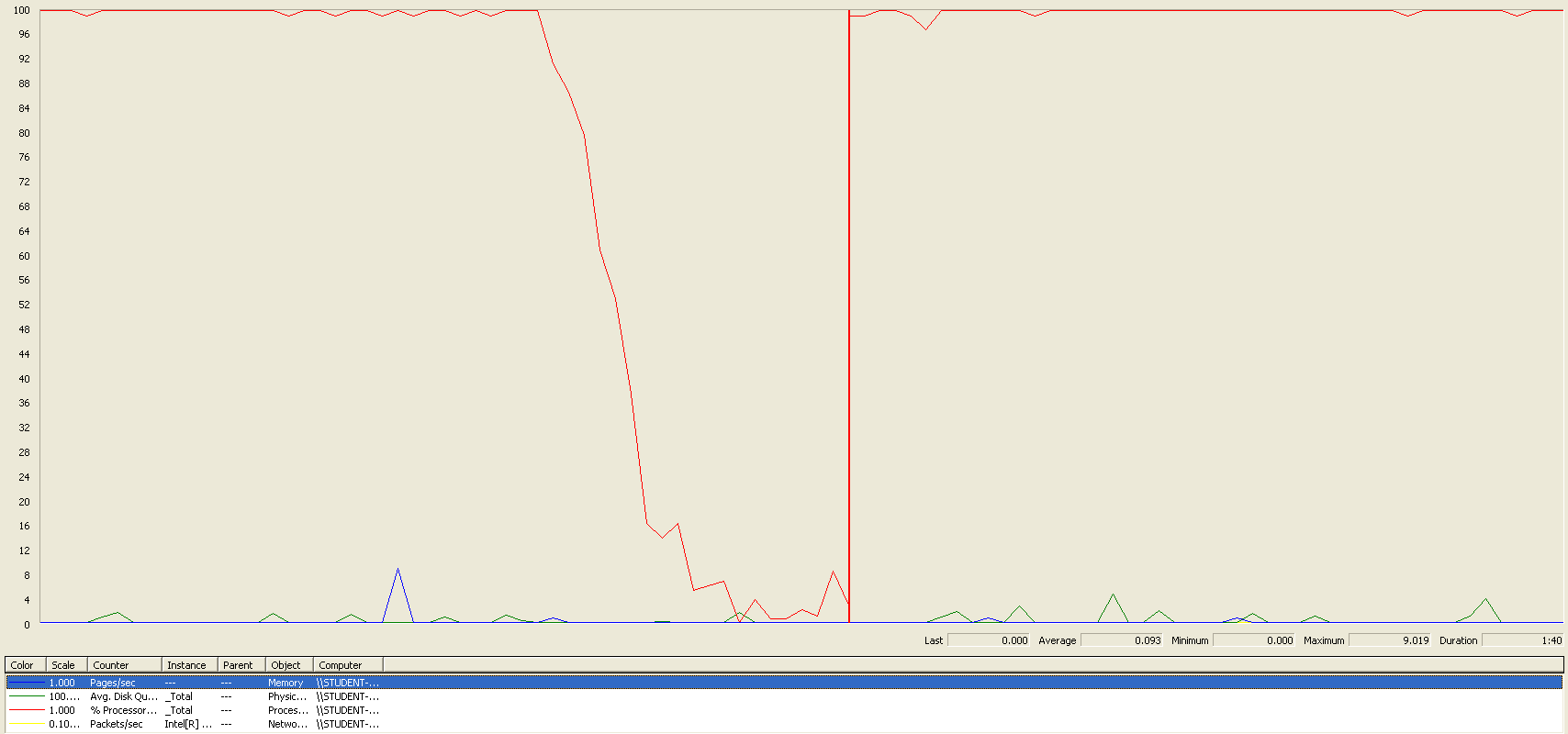
****As we can see in the graph below, there is a bottleneck with the CPU. Probably we could improve the maximum workload of the application if we assign more processors to the virtual machine.

Figure 7. Bottleneck analysis

**Conclusion:** The maximum number of concurrent users supported by this test case is 425 and we could improve it by assigning more CPU’s resources to our system.

**Req. 17.1** – An actor who is authenticated as a hacker must be able to manage his or her curricula which includes listing, showing, creating, updating, and deleting them.

Technical details of the computer on which the test has been executed:

* Ram: 8,192 (1x) MB, DDR3L RAM (1,600 MHz)
* CPU: Intel Core i7-5500U
* Disco duro: 1 TB HDD - 5,400 rpm
* Tarjeta de red: Gigabit Ethernet LAN - 10BASE-T/100BASE-TX/1000BASE-T

**Test case description**: In this test case, we test the entire use case in only one script. This is not ideal, since the tests should as specific as possible. However, since we don’t use a CSV in jMeter to record the data used in the tests, the only way to test the performance of “delete” actions (delete curriculum for example) we need to create an object and then delete it. If we don’t do in this way, a thread can delete an entity and when the following thread try to delete the same entity, it will produce some errors (not related with performance) because is trying to delete a non-existing object in the database. So here it is the test case:

* Log in as hacker
* List his curriculum
* Create a curriculum
* Edit the title of the curriculum created
* Edit the personal data of the curriculum
* Create a position data in the curriculum
* Edit the new position data
* Delete the position data
* Create a new position data (we leave a position data created to make a more realistic test when we remove the curriculum.)
* Create an education data
* Edit the new education data
* Delete the education data
* Create a new education data (we leave an education data created to make a more realistic test when we remove the curriculum.)
* Create a miscellaneous data in the curriculum
* Edit the new miscellaneous data
* Delete the miscellaneous data
* Create a new miscellaneous data (we leave a miscellaneous data created to make a more realistic test when we remove the curriculum.)
* Delete the curriculum
* Log out

**Maximum workload test case.** 90 concurrent users and 20 of loop count:

Figure 8. Configuration

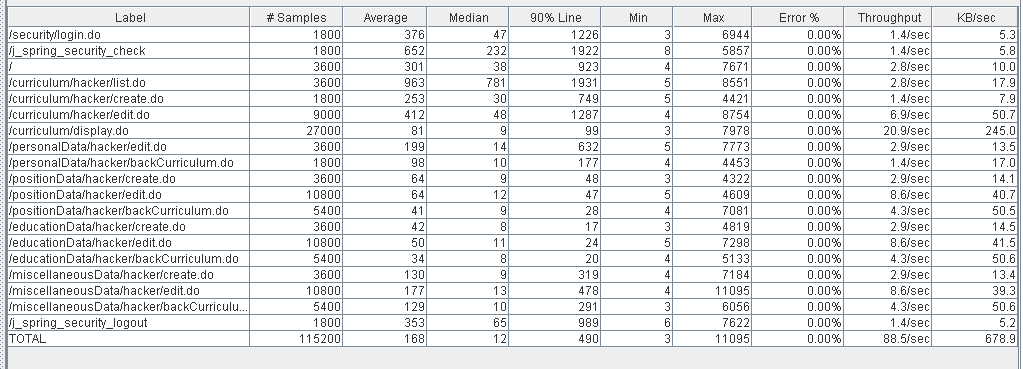
This is the maximum workload of the test case without any crash or excesive delay. As we can see in the picture below, we don’t have any errors and the average time per request is acceptable.

Figure 9. Aggregate report

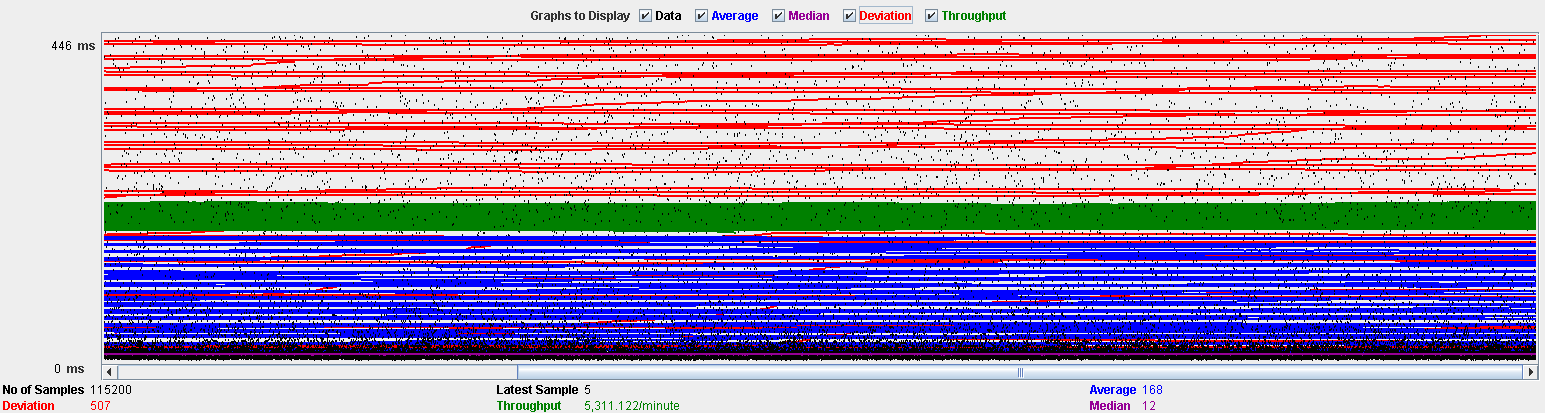


Figure 10. Stabilized output graph report

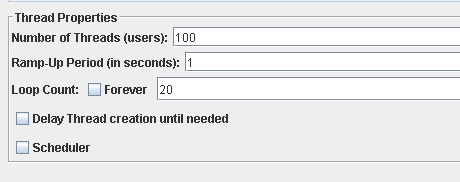
**Overload test case:** 100 concurrent users and 20 of loop count:

Figure 11. Configuration

We don’t have any errors, but the system works with excessive delay.

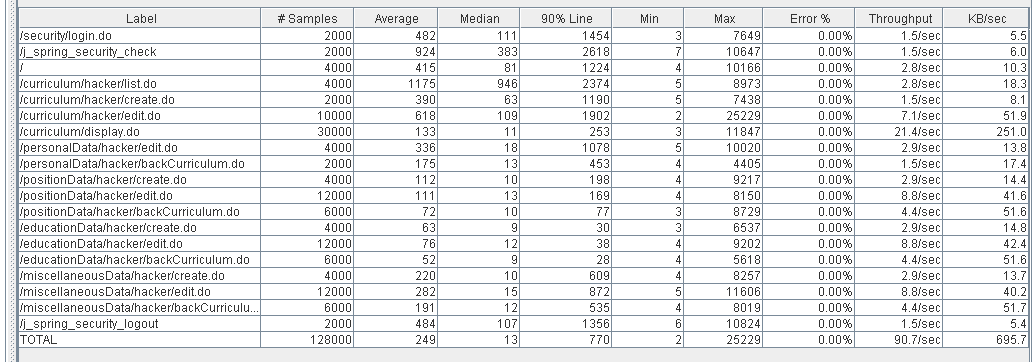
As we can see in the table below, the request with the highest average time is /j\_spring\_security\_check. This URI correspond to an internal implementation of spring, so we cannot improve the performance by refactoring the code.

Figure 12. Aggregate report

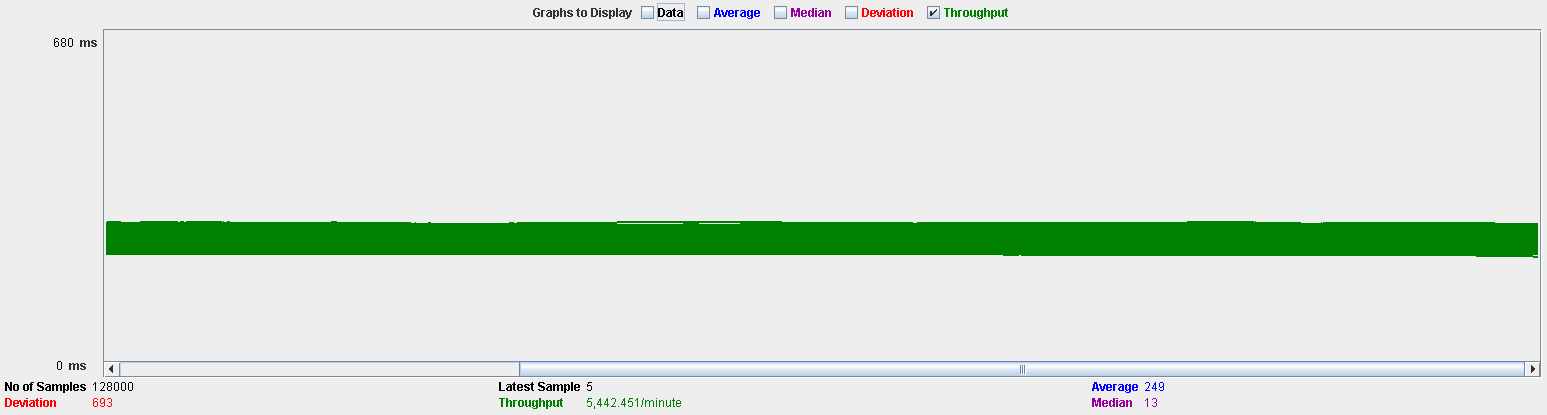


Figure 13. Stabilized output graph report

In the following graphic, we can see that there is a bottleneck in the CPU. We could get a better performance improving CPU’s resources but not too much because, as we can see in the graphic, the CPU is not always working at 100%.

Figure 14. Bottleneck graph

**Conclusion:** The maximum number of concurrent users supported by this test case is 90. We have three alternatives to improve the performance:

1. Assign more CPU’s resources to our system. But as we said previously, this probably won’t do too much.
2. Update the components of our system (tomcat for example). Updated components could have better performance.
3. Improve the Pre-Production environment quality. Using a virtual machine with windows XP could not be the best idea because Windows XP is an outdated operating system and there are much better operating systems for holding a Production environment. Using a virtual machine also reduces the real potential of the hardware of the computer.

**Req. 17.2** – An actor who is authenticated as hacker must be able to manage his or her finder, which involves updating the search criteria, listing its contents and clearing it.

Technical details of the computer on which the test has been executed:

* Ram: 8,192 (1x) MB, DDR3L RAM (1,600 MHz)
* CPU: Intel Core i7-5500U
* Disco duro: 1 TB HDD - 5,400 rpm
* Tarjeta de red: Gigabit Ethernet LAN - 10BASE-T/100BASE-TX/1000BASE-T

**Test case description:**

* Log in as hacker
* Go to Finder view
* Clear the Finder
* Edit Finder with key word “56”, maximum deadline “12/12/2019” and minimum salary “5.0”.
* Log out

**Maximum workload test case.** 250 concurrent users and 20 of loop count:

Figure 15. Configuration

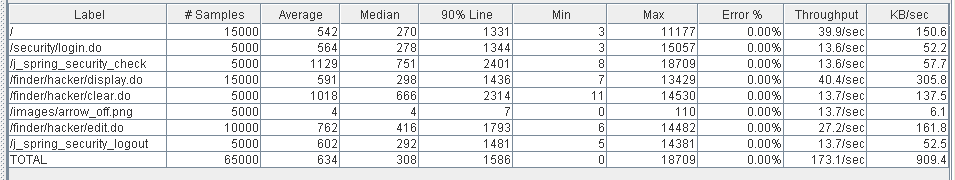
This is the maximum workload of the test case without any crash or excesive delay. As we can see in the picture below, we don’t have any errors and the average time per request is high for some requests, but still acceptable.

Figure 16. Aggregate report

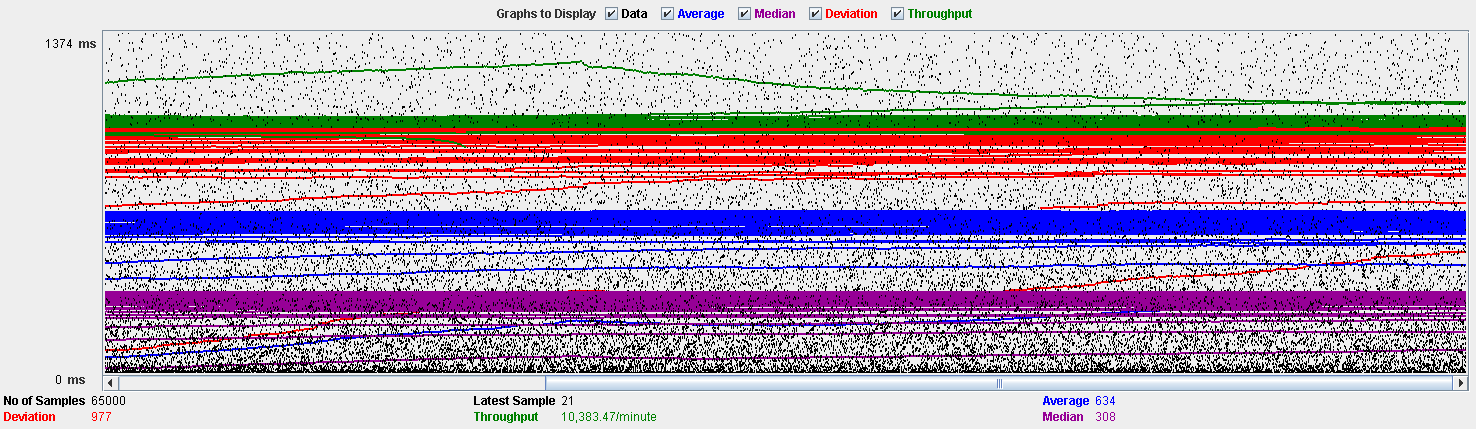
****

Figure 17. Stabilized output graph report

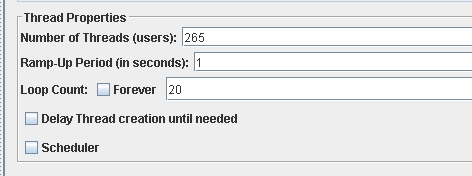
**Overload test case:** 265 concurrent users and 20 of loop count:

Figure 18. Configuration

We don’t have any errors, but the system works with excessive delay.

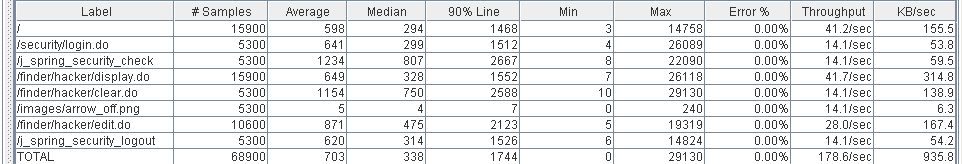
As we can see in the table below, the request with the highest average time is /j\_spring\_security\_check. This URI correspond to an internal implementation of spring, so we cannot improve the performance by refactoring the code.

Figure 19. Aggregate report

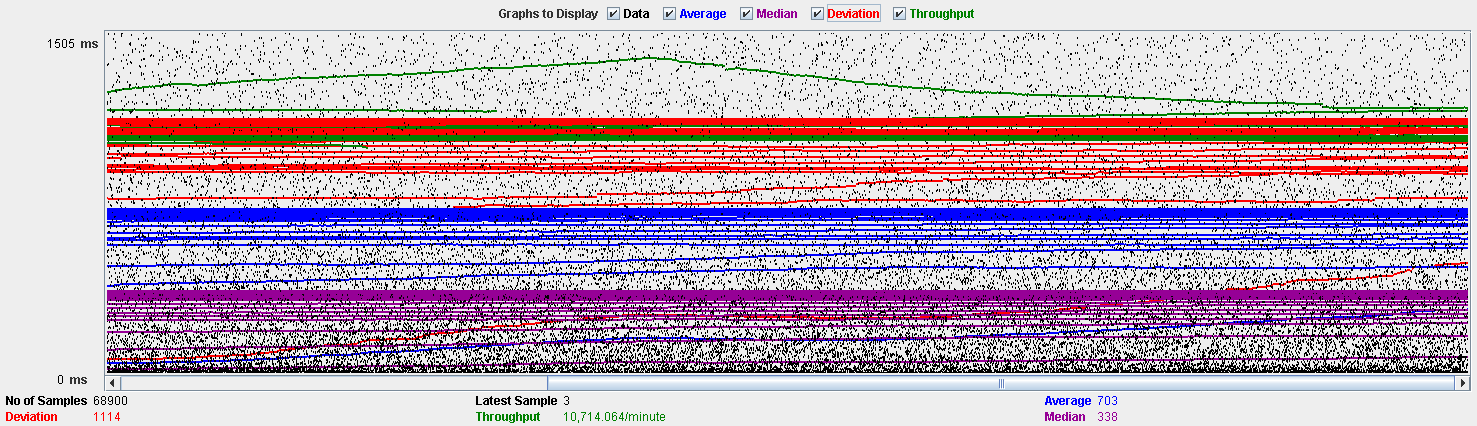


Figure 20. Stabilized output graph report

In the following graphic, we can see that there is a bottleneck in the CPU. We can see also some spikes of the hard drive, but it’s not important since the spikes never reach the 100% of capability. So, we could improve the maximum workload of the application if we assign more processors to the virtual machine.

Figure 21. Bottleneck graph

**Conclusion:** The maximum number of concurrent users supported by this test case is 250. We could improve the performance by assigning more CPU’s resources to our system.