**Req. 14 -** The system must be easy to customise at run time. The customisation includes, but is not limited to: the name of the system (it’s “Acme Hacker Rank” by default); the banner shown at the header (it’s the one available at https://i.imgur.com/7b8lu4b.png by default); the message that is shown on the welcome page (“Welcome to Acme hacker Rank! We’re IT hacker’s favourite job marketplace!” is the default welcome message in English; “¡Bienvenidos a Acme Hacker Rank! ¡Somos el mercado de trabajo favorito de los profesionales de las TICs!” is the default welcome message in Spanish); and the default country code in tele-phone numbers (it’s “+34”by default).

**Req. 19 -** The results of a finder are cached for one hour by default. The administrator should be able to configure that period at will in order to adjust the performance of the system. The minimum time’s one hour and the maximum time’s 24 hours. When a user requests to clear his or her finder, the system must re-compute its results immediately.

**Req. 20 -** The maximum number of results that a finder returns is 10 by default. The administrator should be able to change this parameter in order to adjust the performance of the system. The absolute maximum is 100 results.

**Req. 26 -** The default list of spam words includes “sex”, “viagra”, “cialis”, “one million”, “you’ve been selected”, “Nigeria”, and their corresponding Spanish translations.

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

* Memory RAM: 16 GB DDR4 Memory
* CPU: Intel(R) Core (TM) i7-7500U CPU @ 2.70 GHz (4 CPUs) ~ 2.90 GHz
* Hard Disk: 256 GB SSD + 1000 GB HDD
* Interface network: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter

**Test case description:** first, an user logs in as administrator. Then, the administrator displays customisation. Later, the administrator edits customisation’s parameters. Finally, the administrator logs out.

**Maximum workload test case:** 180 concurrent users, 20 of loop count and 1 of ramp-up period.

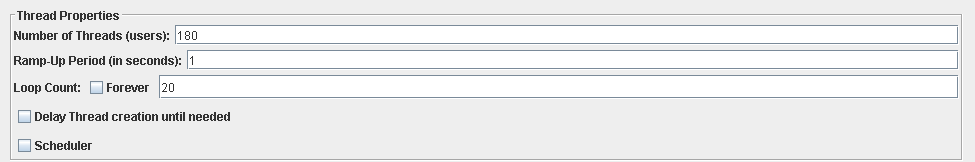


Figure 1 – Configuration

According to the performance test, this is the maximum workload that can be supported by the system without errors and failures or insufficient performance.

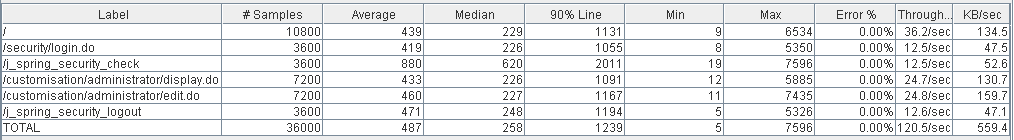


Figure 2 - Aggregate report

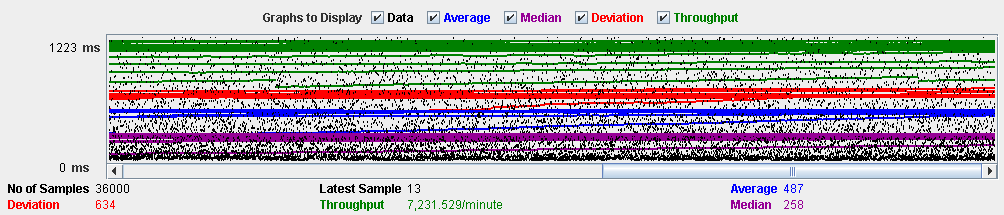


Figure 3 - Graph report

**Overload test case:** 190 concurrent users, 20 of loop count and 1 as ramp-up period.

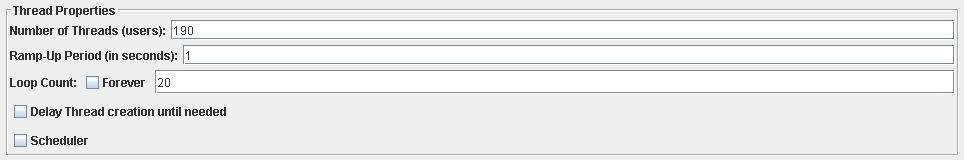


Figure 4 – Configuration

Although there aren’t errors, the average time per request is not acceptable. The request with highest time is “j\_spring\_security\_check”, this is an internal unit of Spring. That’s means that we cannot improve the performance by refactoring the code. Surely, that event is due to a bottleneck component.

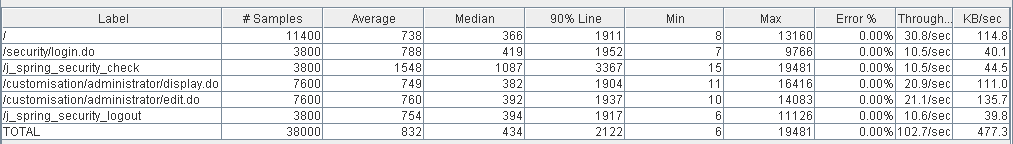


Figure 5 - Aggregate report

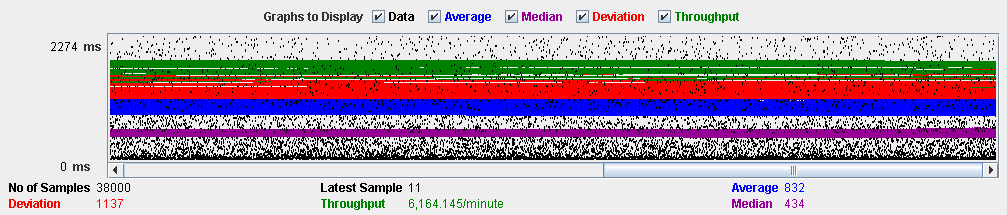


Figure 6 - Graph report

In the following performance analysis, we can appreciate that CPU is saturated. So, there is a bottleneck with CPU. If we increase the units of processor to the virtual machine, the maximum workload would be higher.

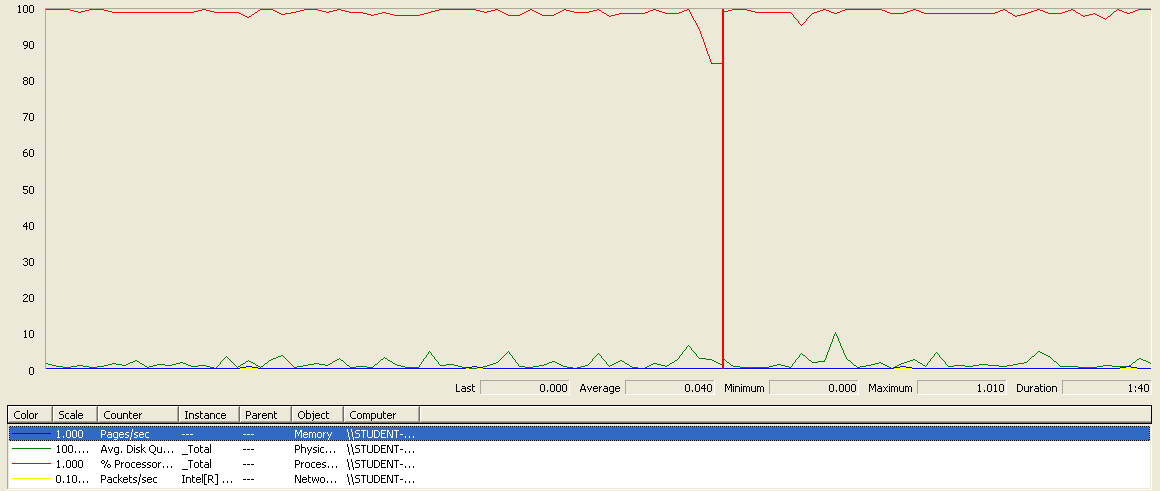
****

Figure 7 - Performance meter

**Conclusion:** The maximum number of concurrent users supported by this test is 180 because the current CPU can handle more workload.

**Req. 21 -** The actors of the system can register their social profiles. The system must store the following data regarding them: a nick, the name of the social network, a link to a profile in that social network.

**Req. 23.1 -** An actor who is authenticated must be able to: Manage his or her social profiles, which includes listing, showing, creating, updating, and deleting them.

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

* Memory RAM: 16 GB DDR4 Memory
* CPU: Intel(R) Core (TM) i7-7500U CPU @ 2.70 GHz (4 CPUs) ~ 2.90 GHz
* Hard Disk: 256 GB SSD + 1000 GB HDD
* Interface network: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter

**Test case description:** first, an authenticated user logs in to the system. Next, he or she goes to his or her profile. Later, the user lists their social profiles and creates a new social profile. Below, the user edits the newly social profile. Then, a social profile is displayed. Immediately, the edited social profile is deleted from the system. Finally, the user is logs out. The entire use case is tested by means of a unique script. It’s not a recommendable strategy.

**Maximum workload test case:** 170 concurrent users, 20 of loop count and 1 as ramp-up period:

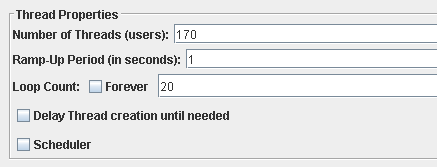


Figure 8 - Configuration

As we can check with the next report, there are not errors and the performance is acceptable.

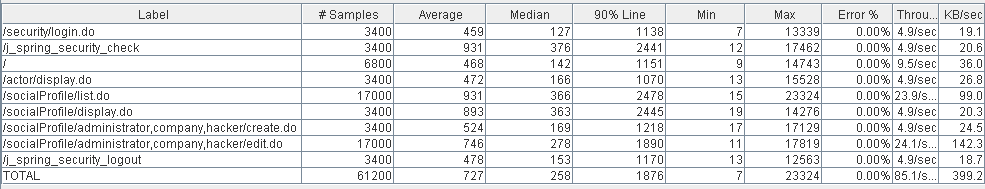


Figure 9 - Aggregate report

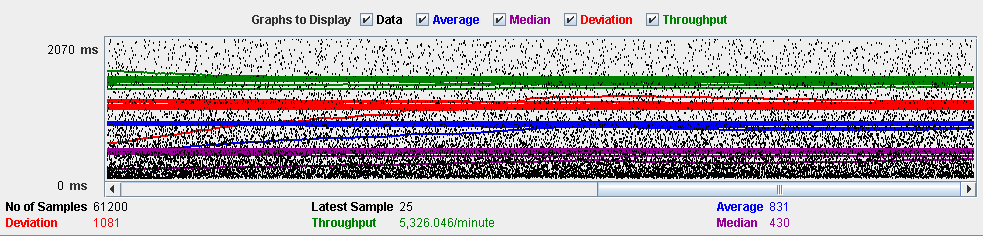


Figure 10 - Graph report

**Overload test case:** 175 concurrent users and 20 of loop count and 1 as ramp-up period:

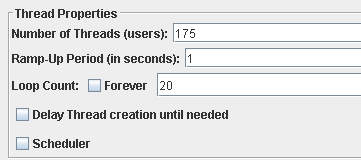


Figure 11 – Configuration

Even though it doesn’t produce any errors, the average time per request is not acceptable. It very possible that a component or several components are bottleneck.

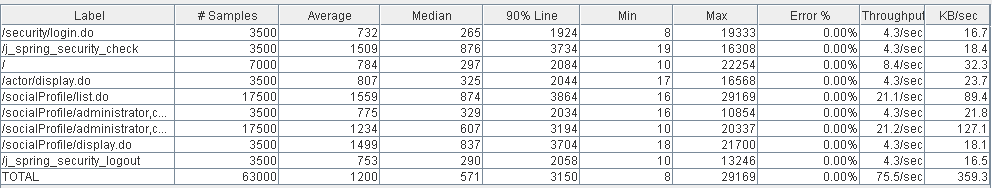


Figure 12 - Aggregate report

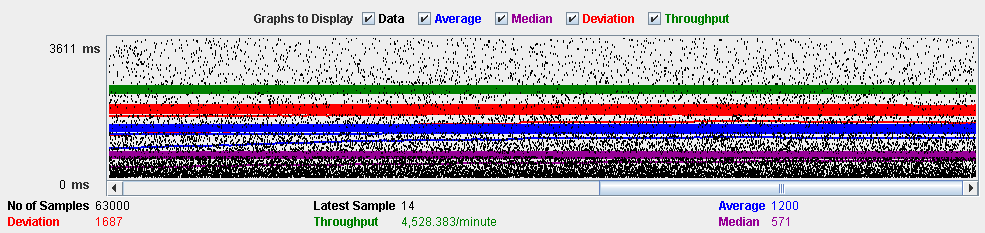


Figure 13 - Graph report

We can see in the figure 14, the CPU is saturated (it’s running at 90%-100% all the time) and hard disk presents frequently spikes at 100%.

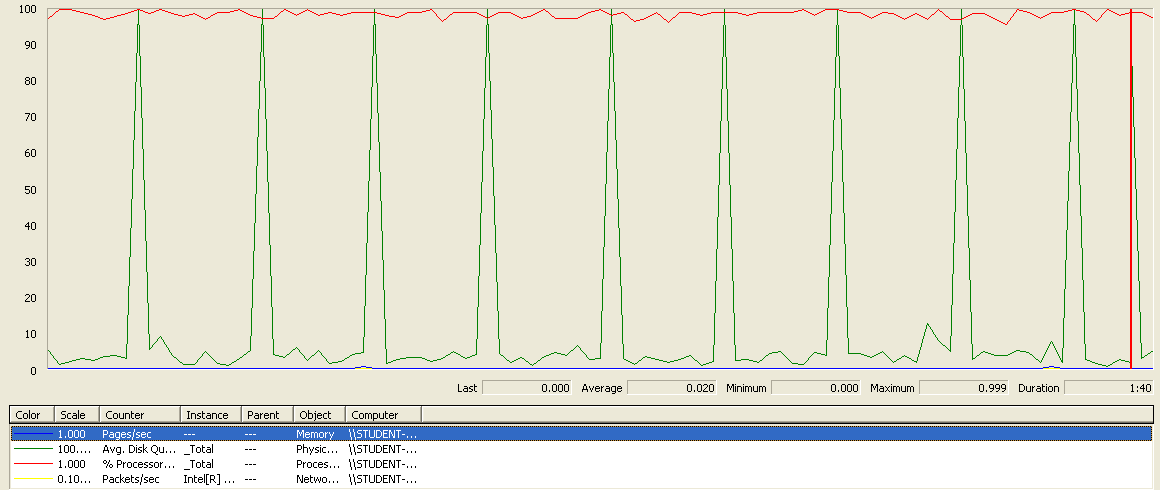


Figure 14 - Performance meter

**Conclusion:** it seems that the bottleneck are CPU and Hard disk: replacing them with more powerful components might boost the system. It’s not necessary to update the interface network and memory.

**Req. 22 -** Every actor has a pool of messages. For every message, the system must keep track of the sender, the recipient, the moment when it was sent, the subject, its body, and some option-al tags.

**Req. 23.2 -** Manage his or her messages, which includes listing them grouped by tag, showing them, sending a message to an actor, deleting a message that he or she’s got. If a message is deleted and it doesn’t have tag “DELETED” then it gets tag “DELETED”, but it’s not actually deleted from the system; if a message with tag “DELETED” is deleted, then it’s actually removed from the system.

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

* Memory RAM: 16 GB DDR4 Memory
* CPU: Intel(R) Core (TM) i7-7500U CPU @ 2.70 GHz (4 CPUs) ~ 2.90 GHz
* Hard Disk: 256 GB SSD + 1000 GB HDD
* Interface network: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter

**Test case description:** first, an authenticated user logs in to the system. Then, the user lists their messages. Immediately, he or she sends a message. Next, a message is displayed. Later, a message is tagged as “DELETED” by the user. After, the message is deleted. Lastly, the user logs out. As happened in social profile’s use case, we use a unique script to test the entire use case.

**Maximum workload test case:** 10 concurrent users, 20 of loop count and 1 of ramp-up period:

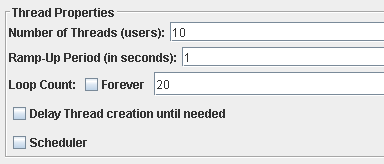


Figure 15 - Configuration

The below reports allow us to check that this is the maximum performance of the system. There are not errors and the performance is good.

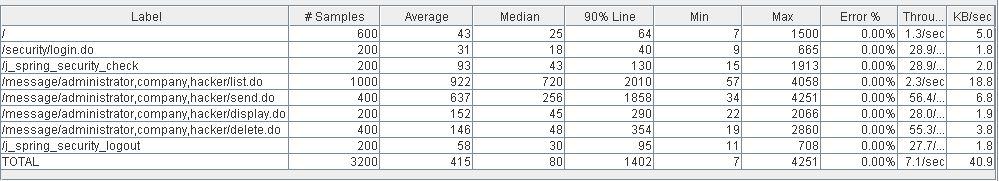


Figure 16 - Aggregate report

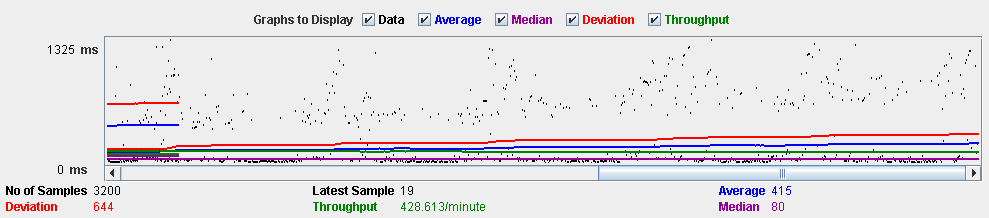


Figure 17 - Graph report

**Overload test case:** 12 concurrent users, 20 of loop count and 1 as ramp-up period:

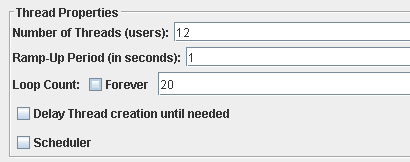


Figure 18 – Configuration

We have not any errors, but the system works with excessive delay. The system takes so much when the messages are listed and a message is sent. Analysing the code, the method MessageService::messageIsSpam is very inefficient due to we must check word by word that the message doesn’t contain a spam word. This method is invoked when a message is sent or when an administrator broadcasts a notification message.

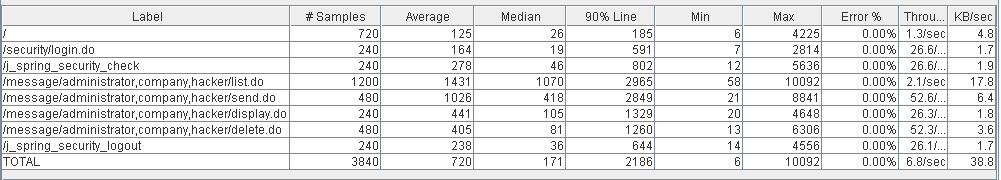


Figure 19 - Aggregate report

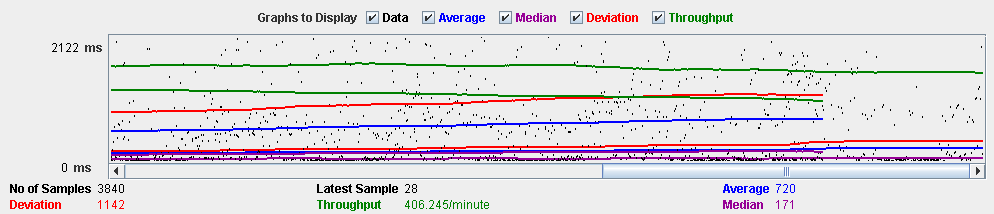


Figure 20 - Graph report

The following picture allow us to check that the CPU works at 100% several time. If we assign more CPU resources, we would get a little improvement. Regarding hard disk, it doesn’t compromise the performance of the system. The other components are idle.

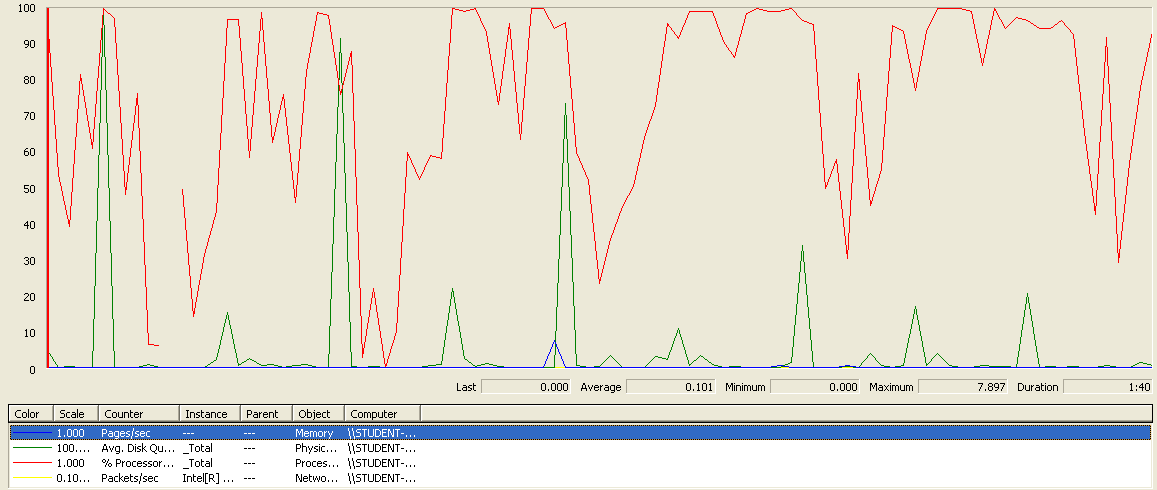


Figure 21 - Performance report

**Conclusion:** we would a little improvement if we replace the current CPU by other more powerful. But we get essential progress if we refactor the code (the method MessageService::messageIsSpam principally).

**Req. 24.1 -** Broadcast a notification message to the actors of the system. The message must have tag “SYSTEM” by default.

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

* Memory RAM: 16 GB DDR4 Memory
* CPU: Intel(R) Core (TM) i7-7500U CPU @ 2.70 GHz (4 CPUs) ~ 2.90 GHz
* Hard Disk: 256 GB SSD + 1000 GB HDD
* Interface network: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter

**Test case description:** first, an administrator logs in to the system. Then, an administrator goes to his or her messages list. Later, an administrator broadcasts a notification message. Finally, the administrator logs out.

**Maximum workload test case:** 11 concurrent users, 20 of loop count and 1 of ramp-up period:

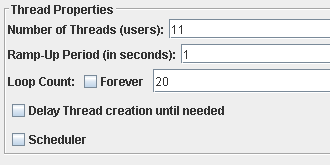


Figure 22 – Configuration

This is the maximum performance of the system without errors and failures or notable delays.

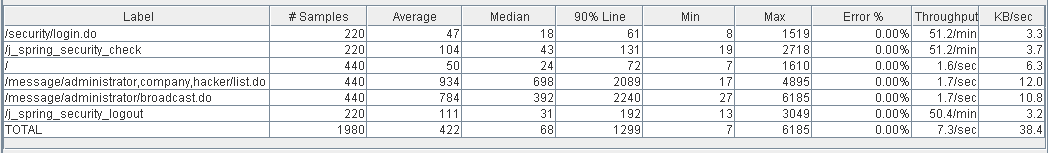


Figure 23 - Aggregate report

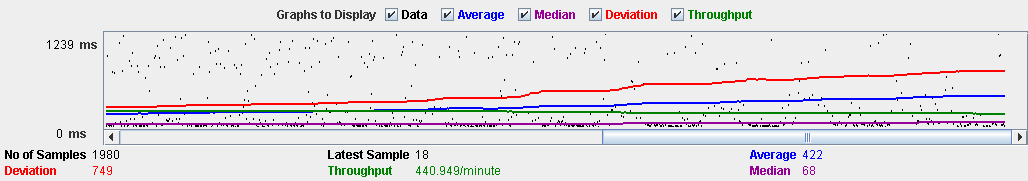


Figure 24 - Graph report

**Overload test case:** 12 concurrent users, 20 of loop count and 1 as ramp-up period:

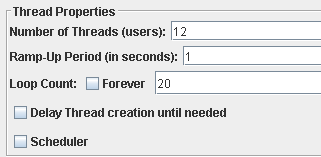


Figure 25 - Configuration

Apparently, there are not errors but the average time per request is not enough good as we can check in the next reports. The system suffers to retrieve the message list of the administrator and when system must send the notification message to all the actors. We have the same problem with the previous performance test (Message’s test performance). The method MessageService::messageIsSpam is not efficient.

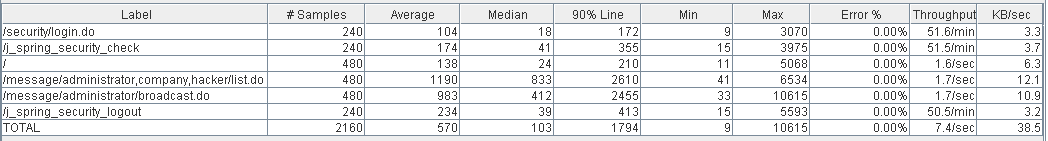


Figure 26 - Aggregate report

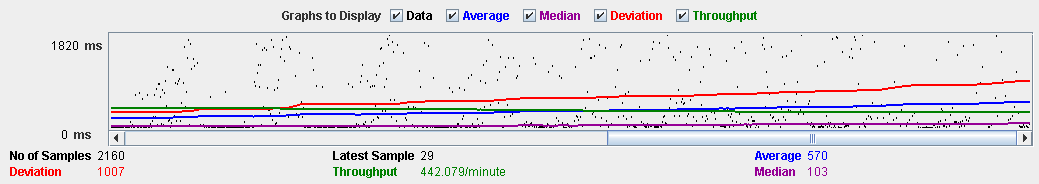


Figure 27 - Graph report

In the next graph, we can see that CPU and hard disk are running at 100 % frequently. If we replace them by other components, we can appreciate a minor progress.



Figure 28 - Performance meter

**Conclusion:** the current CPU and hard disk can’t handle the current workload. if we would use a more powerful CPU and hard disk, the system would support a higher workload. The other components have not to be updated. Besides, if we improve the code (MessageService::messageIsSpam principally), the system would support a higher workload.

**Req. 24.2 -** Launch a process that flags the actors of the system as spammers or not-spammers. A user is considered to be a spammer if at least 10% of the messages that he or she’s sent contain at least one spam word.

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

* Memory RAM: 16 GB DDR4 Memory
* CPU: Intel(R) Core (TM) i7-7500U CPU @ 2.70 GHz (4 CPUs) ~ 2.90 GHz
* Hard Disk: 256 GB SSD + 1000 GB HDD
* Interface network: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter

**Test case description:** first, an administrator logs in to the system. Then, an administrator goes to actor list. Later, an administrator launches the process. Finally, the administrator logs out.

**Maximum workload test case:** 120 concurrent users, 20 of loop count and 1 of ramp-up period:

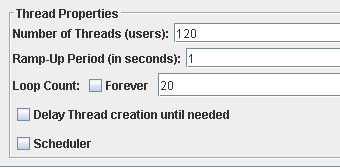


Figure 29 - Configuration

By means of this report, we can appreciate the maximum performance of the system. It doesn’t produce errors and the performance is acceptable.

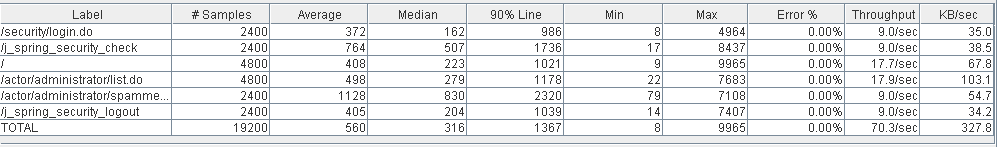


Figure 30 - Aggregate report

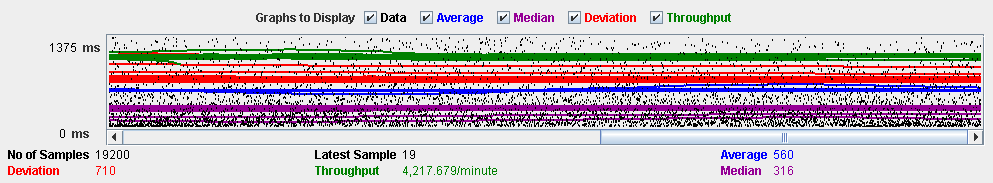


Figure 31 - Graph report

**Overload test case:** 130 concurrent users, 20 of loop count and 1 as ramp-up period:

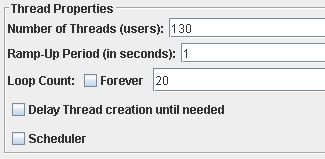


Figure 32 – Configuration

There are not errors but the average time per request is not good. The higher average time corresponds to /actor/administrator/spammer.do. This is an implementation of our application. If we refactor the code, the system would support more concurrent users.

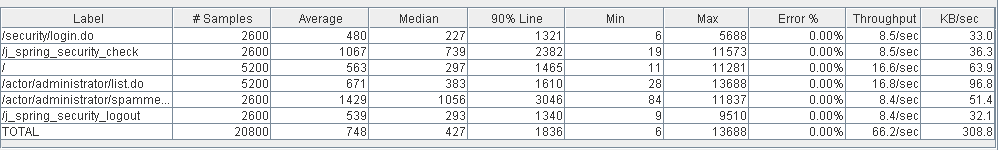


Figure 33 - Aggregate report

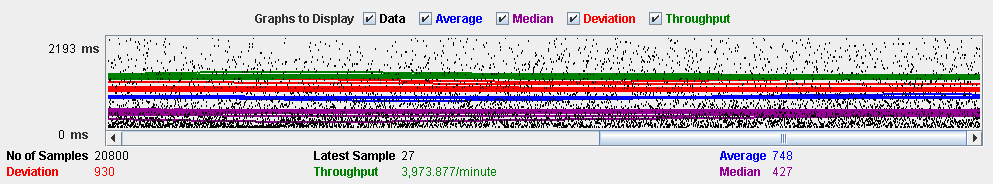


Figure 34 - Graph report

As we can see, CPU is running at 90-100 % all the time. So, CPU can’t handle this workload (bottleneck).

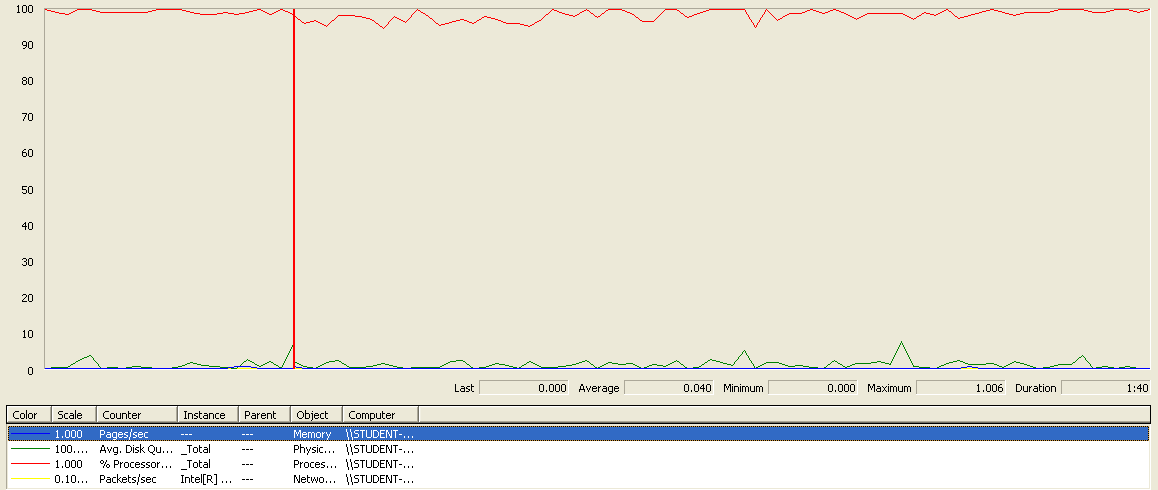


Figure 35 - Performance meter

**Conclusion:** if we increase the CPU resources and we refactor the code, the maximum workload would be higher than the current maximum workload.