25-4-2019

Acme-Hacker Rank

Performance testing

María Jiménez Vega
Álvaro Calle González
Julia García Gallego
Antonio Nolé Anguita
Fernando Manuel Ruiz Pliego

INTRODUCTION

In the following document we will analyze the maximum performance of our system after performing the performance tests using the jmeter tool.

The document consists of all the test cases carried out, for each test case the characteristics of each computer in which the test has been carried out, a series of captures and a conclusion are explained.

It should be noted that although they have been made on different computers all have run the virtual machine "` Pre-Production 1.18.2 ".

In the analysis of the tests we have considered that a time superior to 2500 ms in 90% line is not considered acceptable.

PERFORMANCE TESTING

Req. 7.1 – An actor who is not authenticated must be able to: Register to the system as a company.

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

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

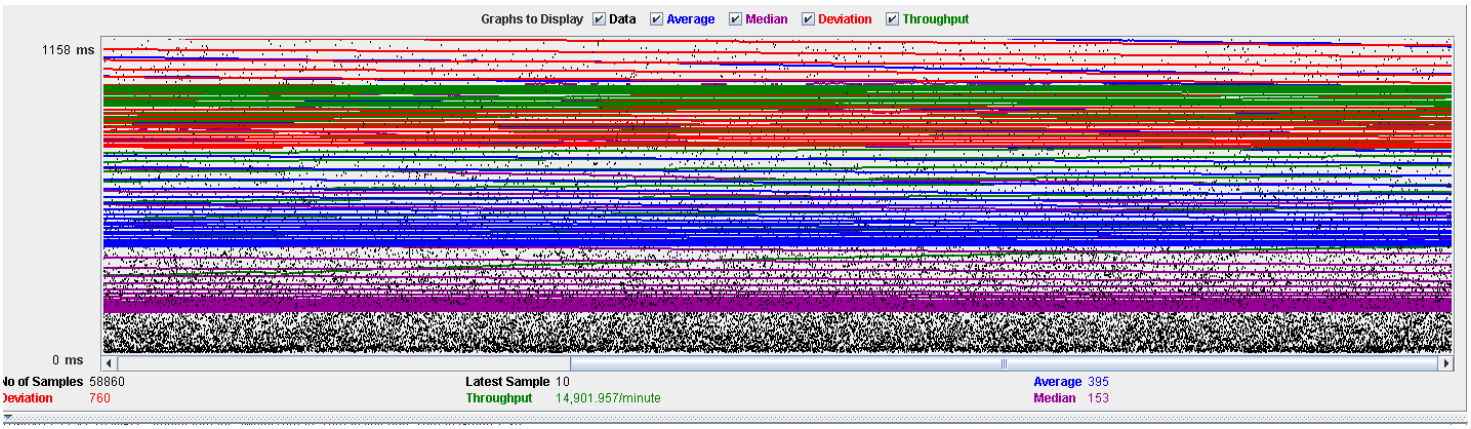
Test case description:

- The user goes to register company view.
- The user completes register company form.
- The user log in the system and then, log out.

Maximum workload test case. 327 concurrent users and 20 of loop count:

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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	19620	348	132	826	2	14804	0.00%	82.8/sec	304.7
/actor/registerCo...	13080	406	152	965	4	11193	0.00%	56.7/sec	527.2
/welcome/index.do	6540	362	134	873	2	9502	0.00%	28.7/sec	103.7
/security/login.do	6540	344	130	862	2	14138	0.00%	28.9/sec	110.4
/j_spring_security...	6540	667	328	1700	7	11122	0.00%	29.0/sec	114.6
/j_spring_security...	6540	322	133	797	3	13072	0.00%	29.3/sec	112.4
TOTAL	58860	395	153	972	2	14804	0.00%	248.4/sec	1237.9

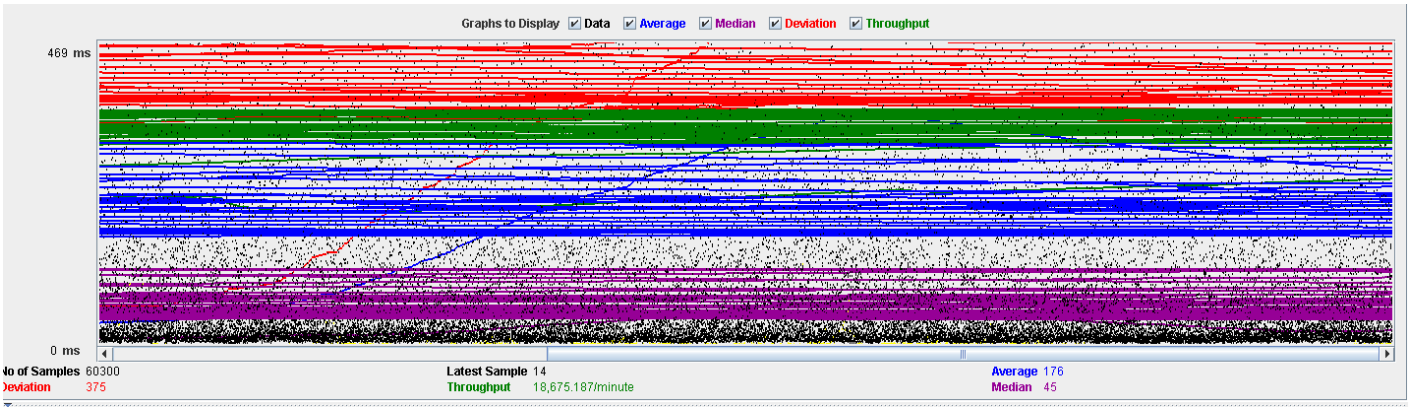


Overload test case: 335 concurrent users and 20 of loop count:

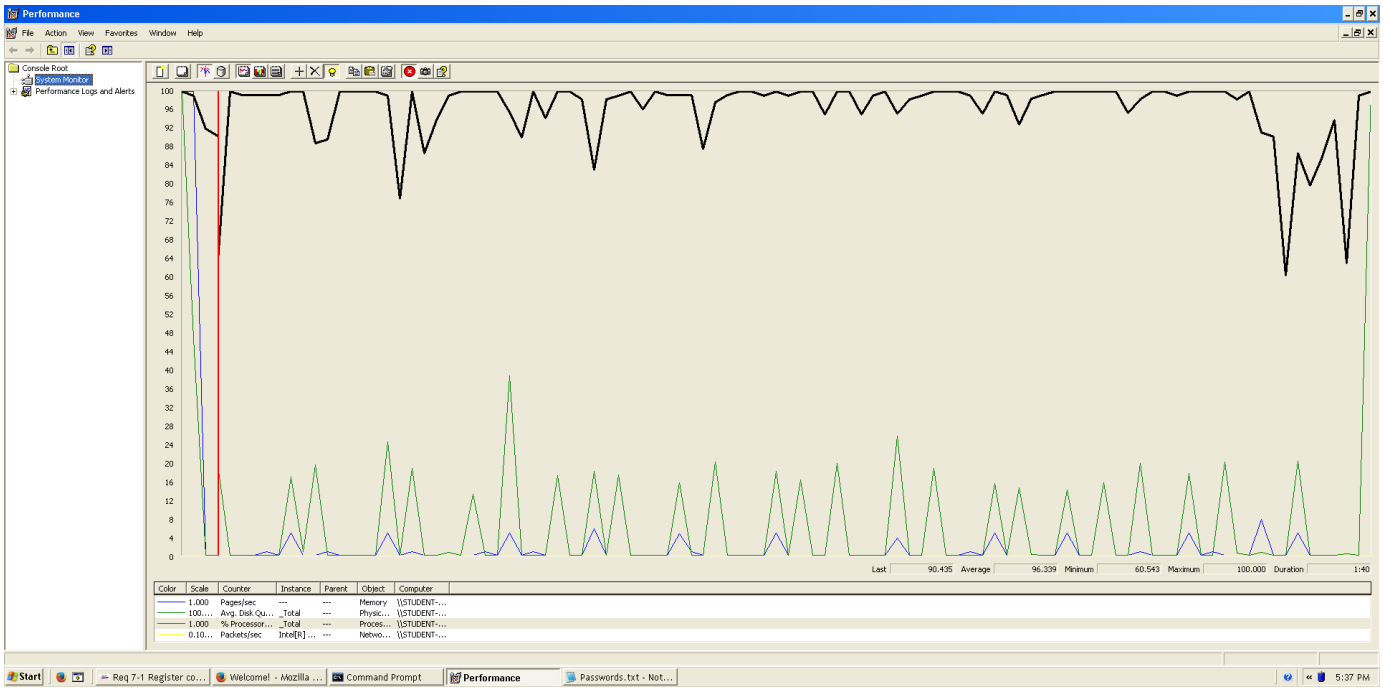
Even though the average time per request is still acceptable, it begins to produce some errors as we can see in the following picture. The errors are always the same: I/O exception (java.net.BindException) caught when connecting to the target host:

Address already in use: connect 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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	20100	152	37	402	0	9286	0.89%	103.8/sec	380.0
/actor/registerCompany....	13400	174	48	457	0	6705	0.95%	71.4/sec	658.7
/welcome/index.do	6700	163	39	458	0	5445	1.13%	36.0/sec	129.4
/security/login.do	6700	145	31	383	0	12897	1.12%	35.9/sec	136.5
/j_spring_security_check	6700	305	107	826	0	11611	1.07%	36.0/sec	141.4
/j_spring_security_logout	6700	169	45	450	0	5668	0.97%	36.0/sec	137.5
TOTAL	60300	176	45	474	0	12897	0.98%	311.3/sec	1541.3



As we can see in the graph below, CPU is almost working 100% but there're moments that descend, so we can get more number of concurrent users assigning more CPU's resources and try to improve our code to try to equalizer the CPU working around 100%.



Conclusion: The maximum number of concurrent users supported by this test case is 327 and we could improve it by assigning more CPU's resources to our system and implement our code more efficient.

Req. 7.1 – An actor who is not authenticated must be able to: Register to the system as a hacker.

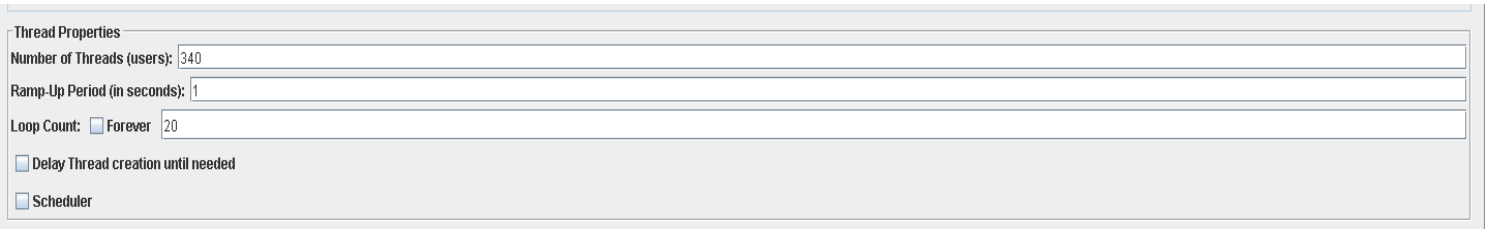
Technical details of the computer on which the test has been executed:

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

Test case description:

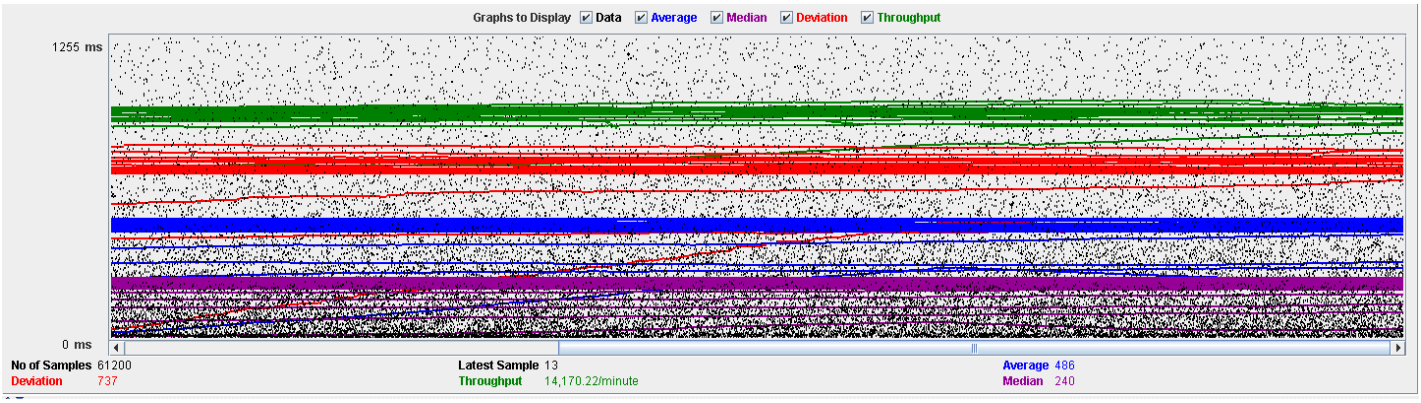
- The user goes to register hacker view.
- The user completes register hacker form.
- The user log in the system and then, log out.

Maximum workload test case. 340 concurrent users and 20 of loop count:

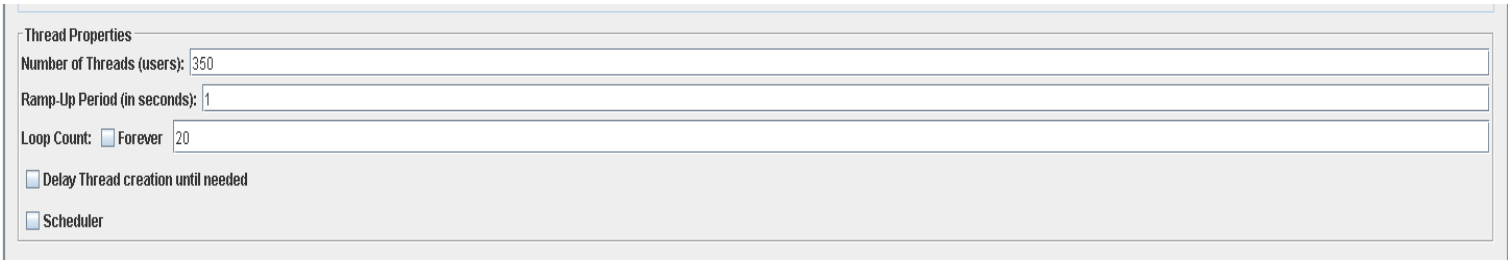


This is the maximum workload of the test case without any crash or excesive delay. As we can see in the picture below, we don't have any errors and the average time per request is pretty acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	20400	415	197	1021	3	11263	0.00%	78.7/sec	297.0
/actor/registerHacker.do	13600	467	236	1143	5	12440	0.00%	53.7/sec	482.0
/welcome/index.do	6800	440	218	1081	3	10078	0.00%	27.0/sec	97.8
/security/login.do	6800	438	212	1041	3	10631	0.00%	27.0/sec	103.3
/j_spring_security_check	6800	861	544	1969	8	12133	0.00%	27.0/sec	114.3
/j_spring_security_logout	6800	456	214	1122	3	10953	0.00%	27.1/sec	103.9
TOTAL	61200	486	240	1197	3	12440	0.00%	236.2/sec	1174.8



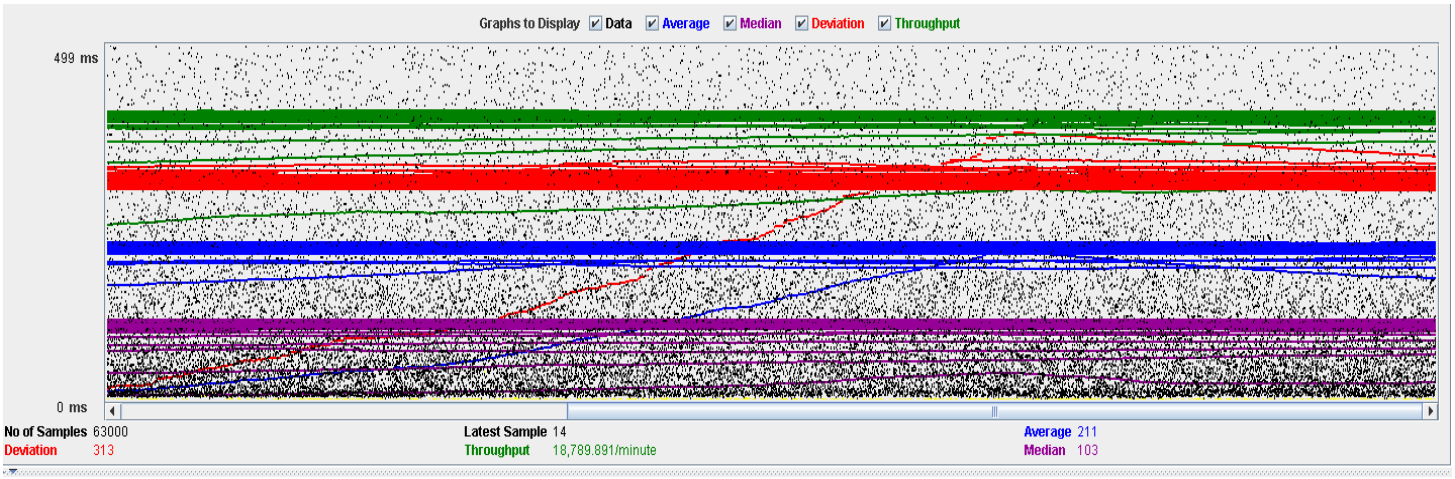
Overload test case: 350 concurrent users and 20 of loop count:



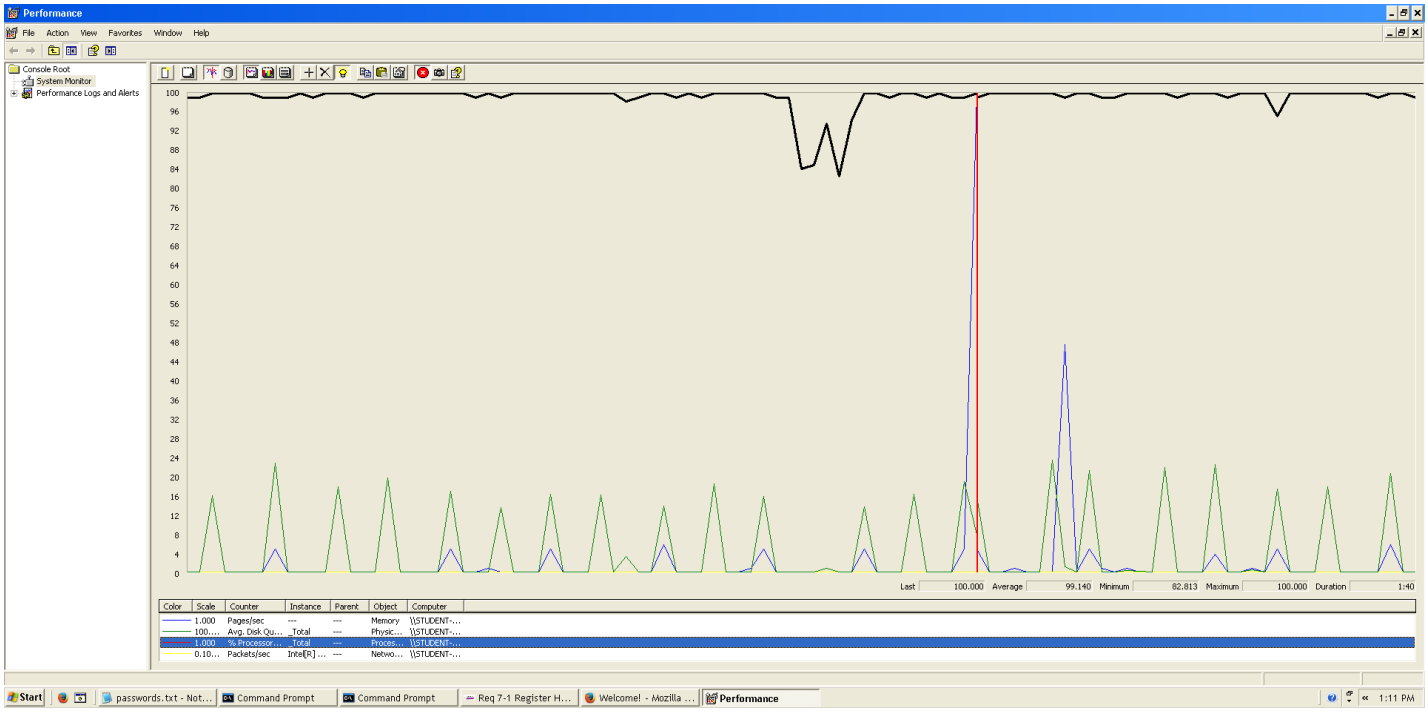
Even though the average time per request is still acceptable, it begins to produce some errors as we can see in the following picture. The errors are always the same: I/O exception (java.net.BindException) caught when connecting to the target host:

Address already in use: connect 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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	21000	181	84	450	0	6796	1.47%	104.4/sec	390.5
/actor/registerHacker.do	14000	208	103	503	0	4169	1.60%	71.8/sec	635.5
/welcome/index.do	7000	197	95	495	0	3214	1.70%	36.1/sec	129.6
/security/login.do	7000	182	89	461	0	3673	1.80%	36.2/sec	136.9
/j_spring_security_check	7000	363	231	837	0	4978	1.56%	36.1/sec	151.2
/j_spring_security_logout	7000	195	99	479	0	3061	1.63%	36.2/sec	137.6
TOTAL	63000	211	103	524	0	6796	1.59%	313.2/sec	1541.0



As we can see in the graph below, there is a bottleneck with the CPU. We could improve CPU's resources because, as we can see it almost works at 100%.



Conclusion: The maximum number of concurrent users supported by this test case is 340 and we could improve it by assigning more CPU's resources to our system.

Req. 7.1 – An actor who is not authenticated must be able to listing available positions.

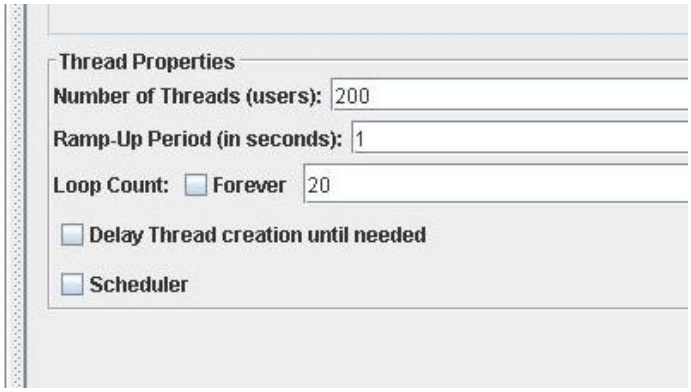
Technical details of the computer on which the test has been executed:

- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

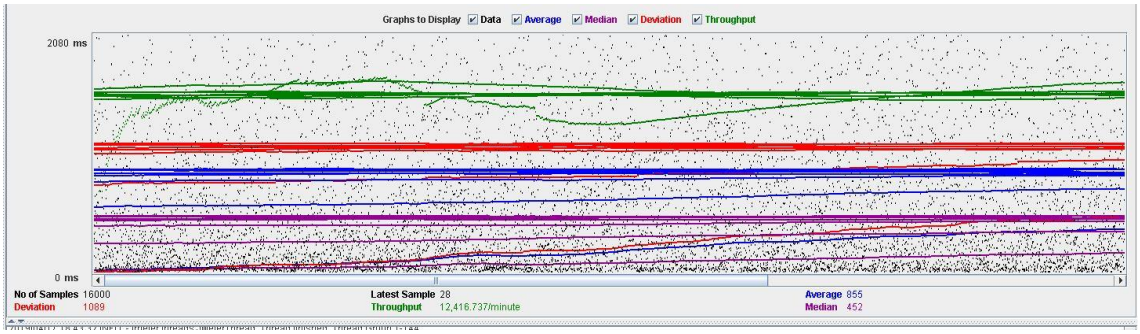
Test case description:

- The user goes to available list in the menu.
- The user display the position.
- Return to homepage.

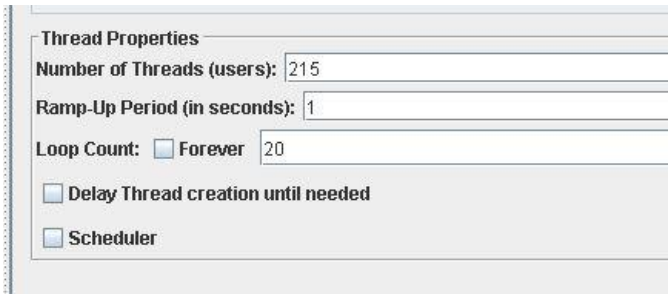
Maximum workload test case. 200 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/position/availableList.do	4000	843	465	2125	0	12949	0.00%	51.8/sec	291.9
/position/display.do	4000	858	444	2223	0	9737	0.00%	51.8/sec	215.7
/position/list.do	4000	880	448	2358	0	11090	0.00%	51.8/sec	269.0
/	4000	837	450	2204	0	11441	0.00%	51.8/sec	187.5
TOTAL	16000	855	452	2221	0	12949	0.00%	206.9/sec	963.3

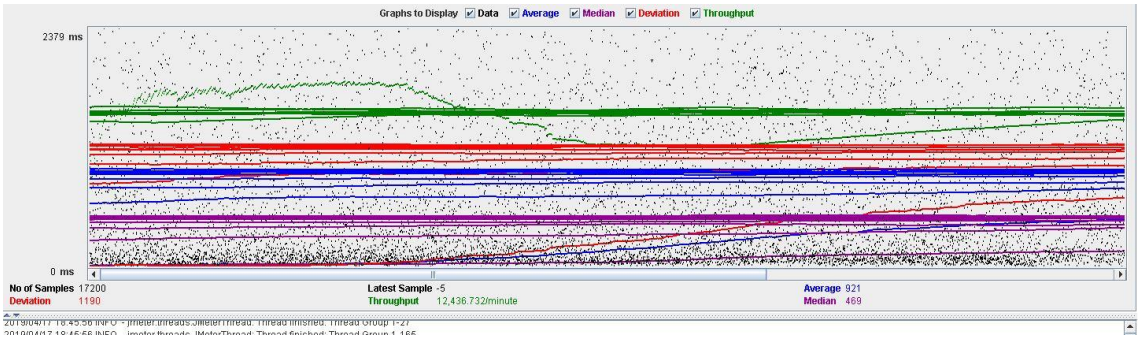


Overload test case:215 concurrent users and 20 of loop count:

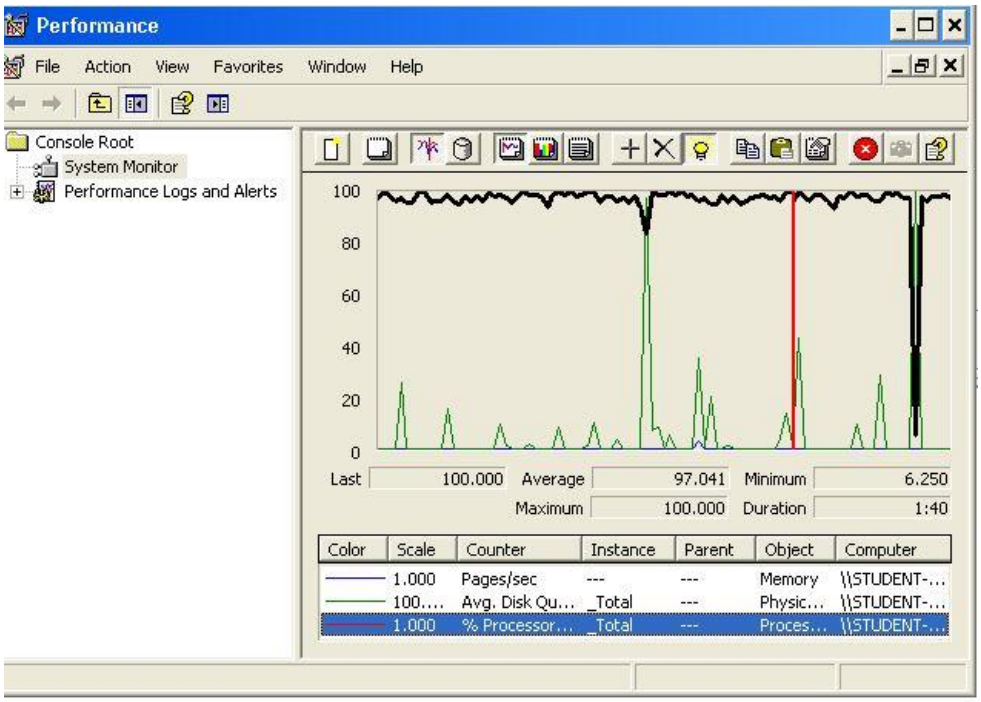


As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throu
/position/availableList.do	4300	985	520	2639	2	11567	0.00%	
/position/display.do	4300	930	474	2409	0	11973	0.00%	
/position/list.do	4300	916	472	2393	1	16035	0.00%	
/	4300	853	416	2251	0	10338	0.00%	
TOTAL	17200	921	469	2432	0	16035	0.00%	



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.



Conclusion: The maximum number of concurrent users supported by this test case is 200 and we could improve it by assigning more CPU's resources to our system.

Req. 7.3 – An actor who is not authenticated must be able to: List the companies available and navigate to the corresponding positions.

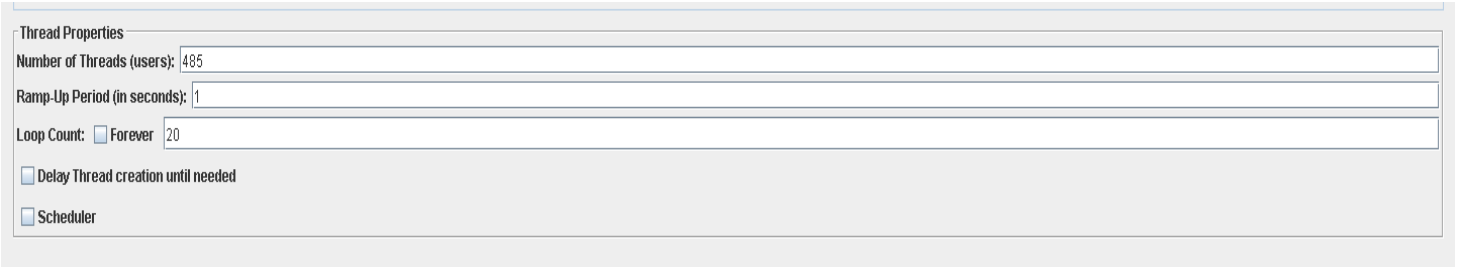
Technical details of the computer on which the test has been executed:

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

Test case description:

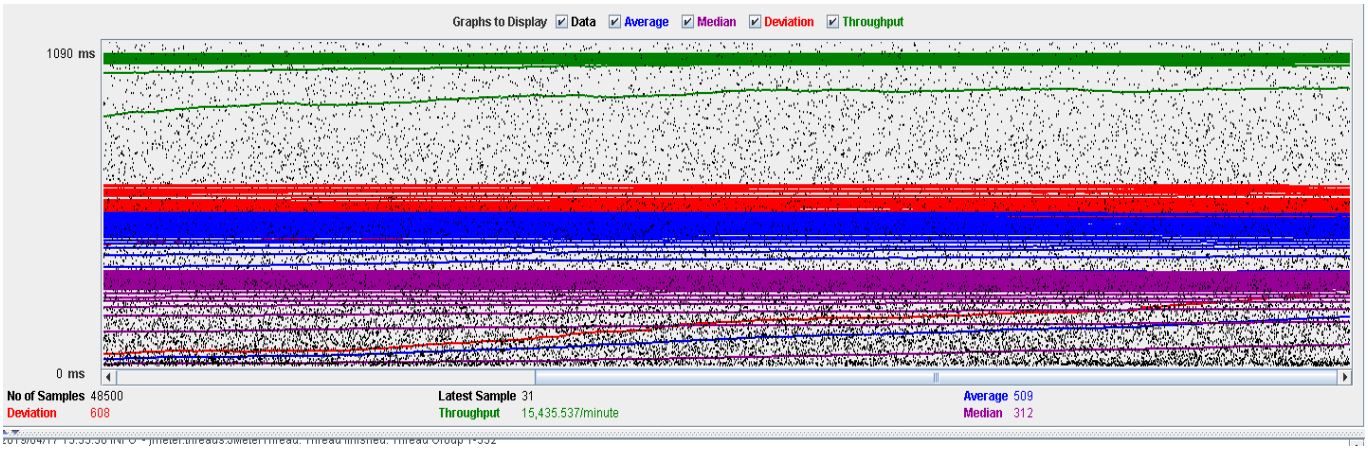
- The user goes to company list view.
- The user display a company from the list.
- The user display de positions of the company.

Maximum workload test case. 485 concurrent users and 20 of loop count:

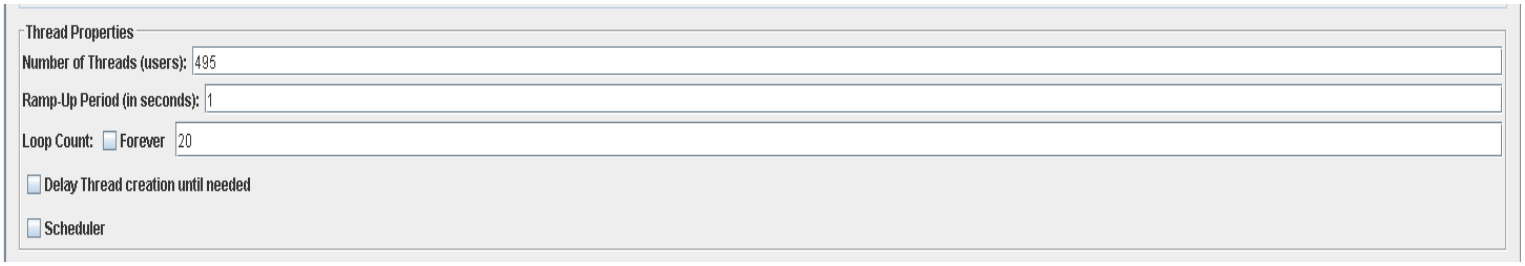


This is the maximum workload of the test case without any crash or excesive delay. As we can see in the picture below, we don't have any errors and the average time per request is pretty acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	9700	319	165	792	2	5013	0.00%	53.3/sec	196.7
/company/list.do	9700	324	168	815	4	8192	0.00%	53.2/sec	224.6
/actor/display.do	9700	618	420	1391	7	10902	0.00%	53.2/sec	201.2
/position/list.do	9700	641	436	1442	6	7646	0.00%	53.2/sec	201.5
/position/display.do	9700	642	444	1453	5	8184	0.00%	53.1/sec	201.1
TOTAL	48500	509	312	1211	2	10902	0.00%	257.3/sec	991.4

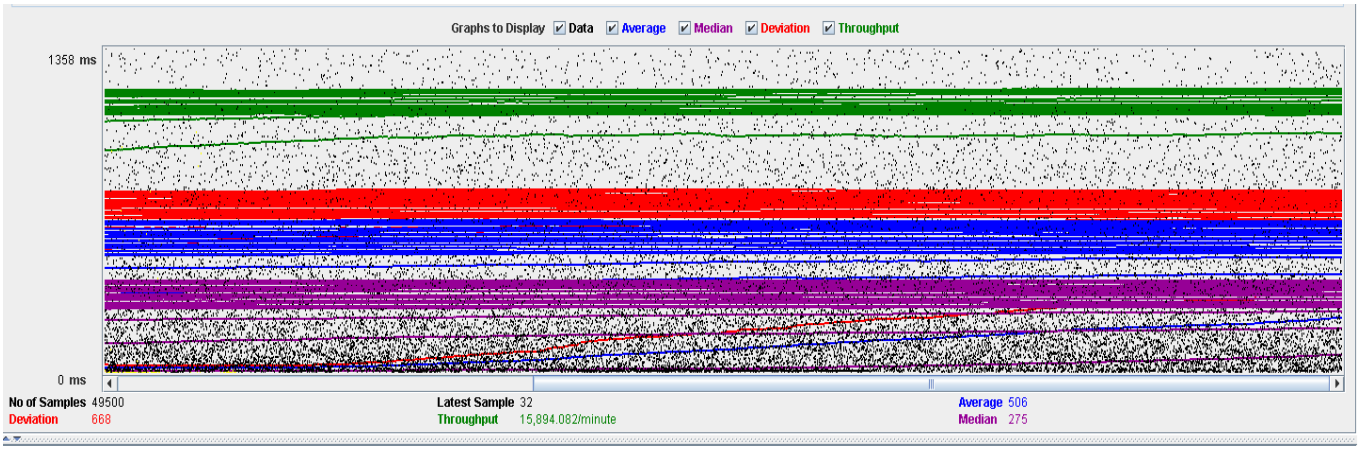


Overload test case: 495 concurrent users and 20 of loop count:

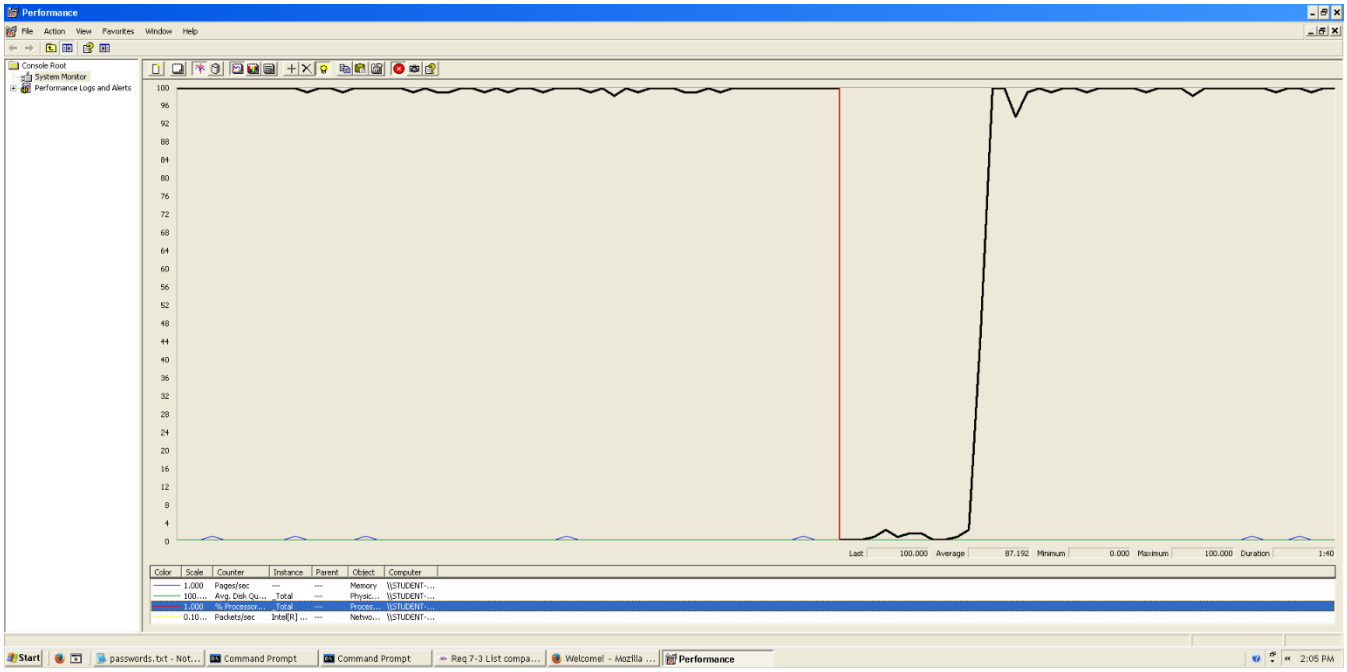


Even though the average time per request is still acceptable, it begins to produce some errors as we can see in the following picture. The errors are always the same: I/O exception (java.net.SocketException) caught when processing request: Connection reset by peer: socket write error This exception is not related with the implementation of our application, but with tomcat. The system can't handle this number of concurrent users properly.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	9900	311	137	793	0	6726	0.38%	54.8/sec	201.9
/company/list.do	9900	325	152	818	1	7748	0.34%	54.8/sec	230.8
/actor/display.do	9900	626	386	1452	1	10292	0.43%	54.8/sec	206.9
/position/list.do	9900	648	402	1522	0	7493	0.55%	54.7/sec	206.4
/position/display.do	9900	622	383	1483	0	9519	0.52%	54.7/sec	206.6
TOTAL	49500	506	275	1265	0	10292	0.44%	264.9/sec	1018.6



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 but not improve it so much because we can see it doesn't work at 100% always.



Conclusion: The maximum number of concurrent users supported by this test case is 485 and we could improve it by assigning more CPU's resources to our system.

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
- Hard disk: 1 TB HDD - 5,400 rpm
- Interface network: Gigabit Ethernet LAN - 10BASE-T/100BASE-TX/1000BASE-T

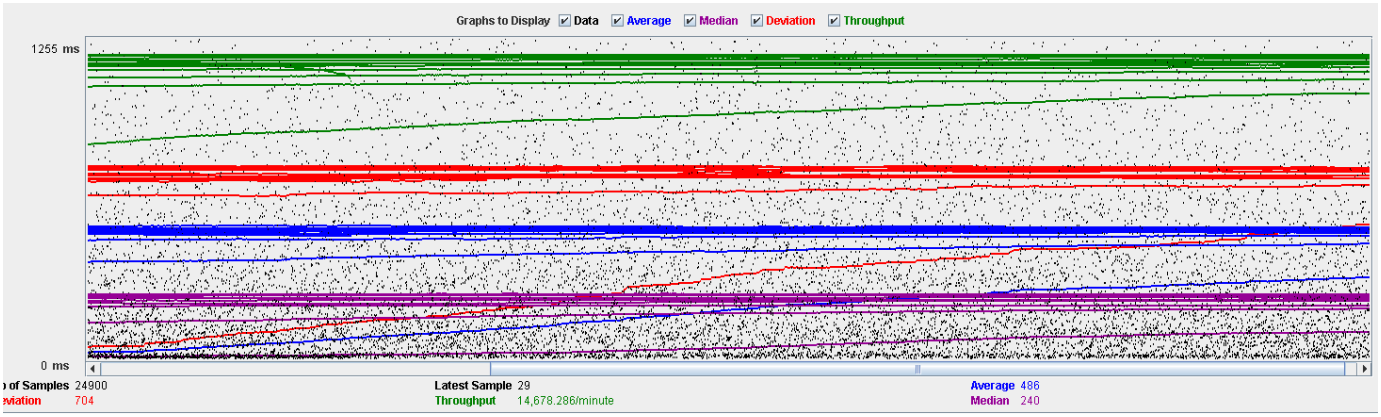
Test case description:

- 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:

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

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	8300	467	219	1197	3	9278	0.00%	83.9/sec	309.7
/position/search.do	16600	496	252	1238	6	9931	0.00%	165.2/sec	944.6
TOTAL	24900	486	240	1225	3	9931	0.00%	244.6/sec	1233.8

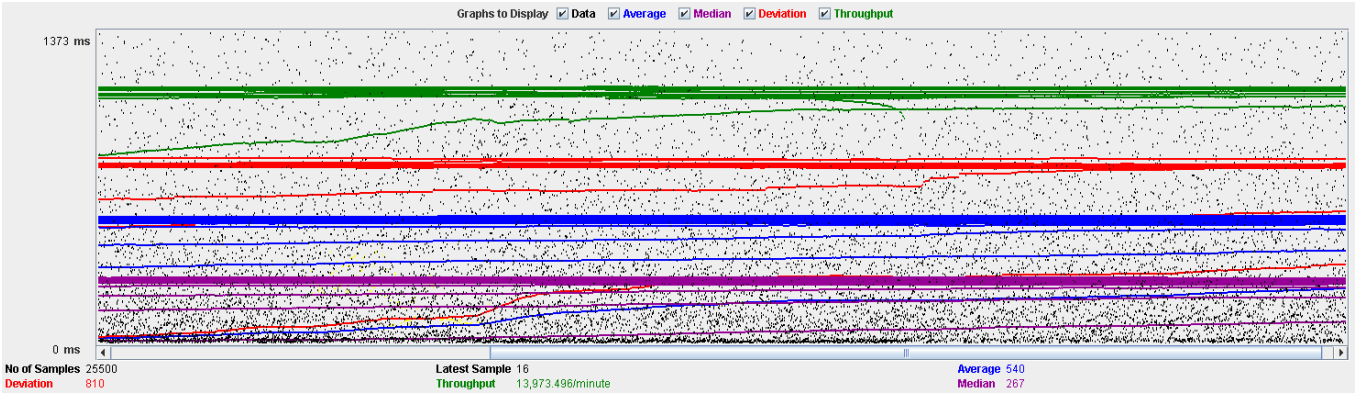


Overload test case: 425 concurrent users and 20 of loop count:

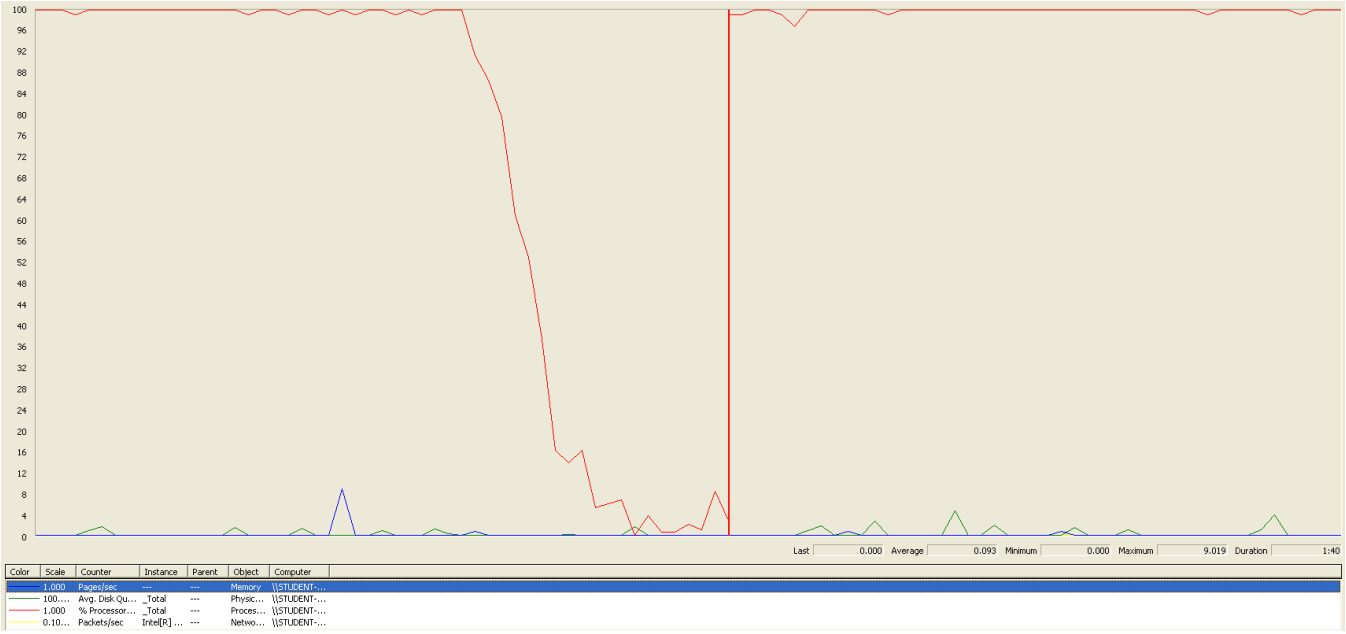
Even though the average time per request is still acceptable, it begins to produce some errors as we can see in the following picture. The errors are always the same: I/O exception (java.net.SocketException) caught when processing request: Connection reset by peer: socket write error

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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	8500	528	253	1311	3	11126	0.79%	79.8/sec	293.7
/position/search.do	17000	546	273	1340	6	12513	0.54%	157.4/sec	896.9
TOTAL	25500	540	267	1330	3	12513	0.62%	232.9/sec	1170.5



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.



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. 8.2 – An actor who is authenticated must be able to: Edit his or her personal data (administrator).

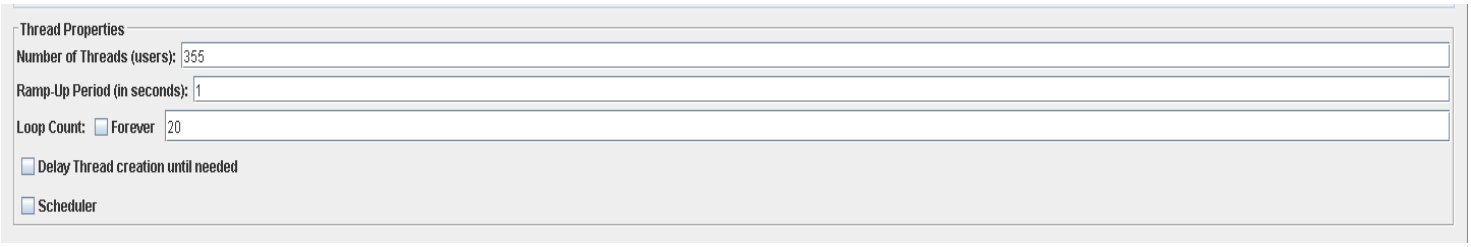
Technical details of the computer on which the test has been executed:

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

Test case description:

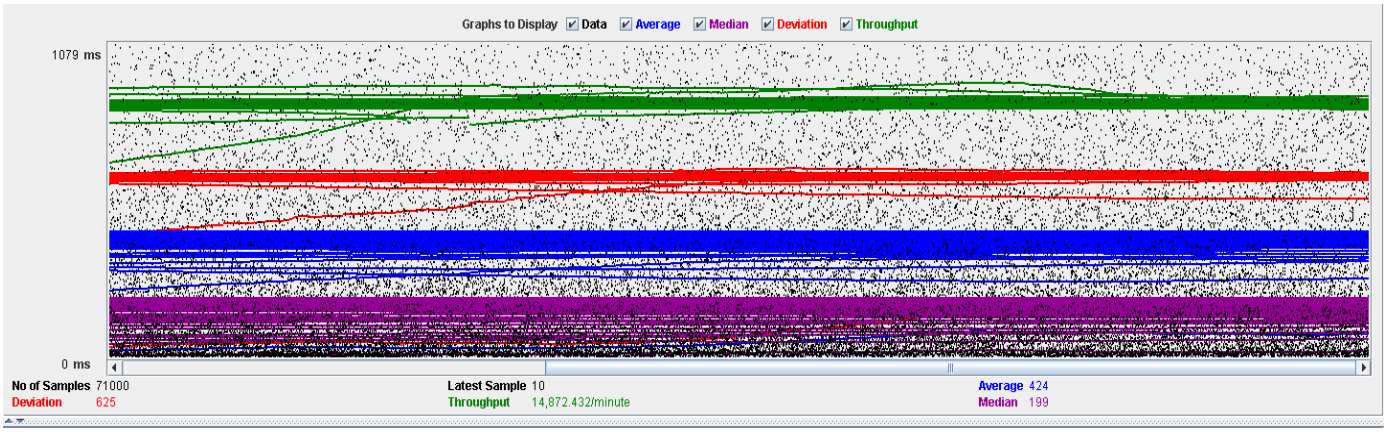
- The user log in the system.
- The user goes to his/her profile.
- The user goes to personal edit form.
- Change any field.
- Press save button save changes.
- Log out the system.

Maximum workload test case. 355 concurrent users and 20 of loop count:



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.

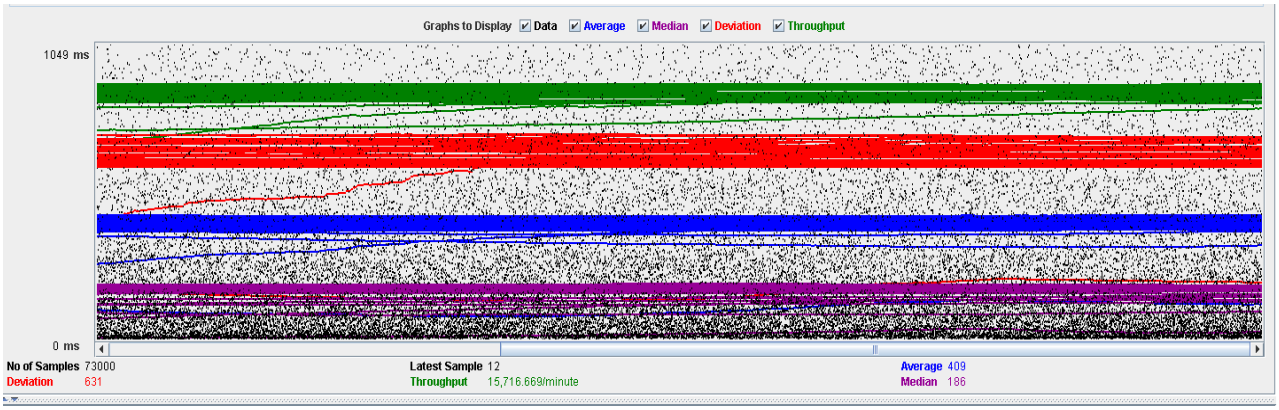
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
	21300	333	152	857	2	9009	0.00%	74.4/sec	282.5
security/login.do	7100	317	135	840	2	6504	0.00%	25.6/sec	98.1
j_spring_security_check	7100	586	343	1421	5	8562	0.00%	25.6/sec	110.4
actor/display.do	14200	370	169	929	4	8865	0.00%	50.5/sec	280.1
actor/administrator,co...	14200	605	325	1534	5	10074	0.00%	50.8/sec	333.9
j_spring_security_logout	7100	390	182	1006	4	8799	0.00%	25.5/sec	98.0
TOTAL	71000	424	199	1093	2	10074	0.00%	247.9/sec	1180.6



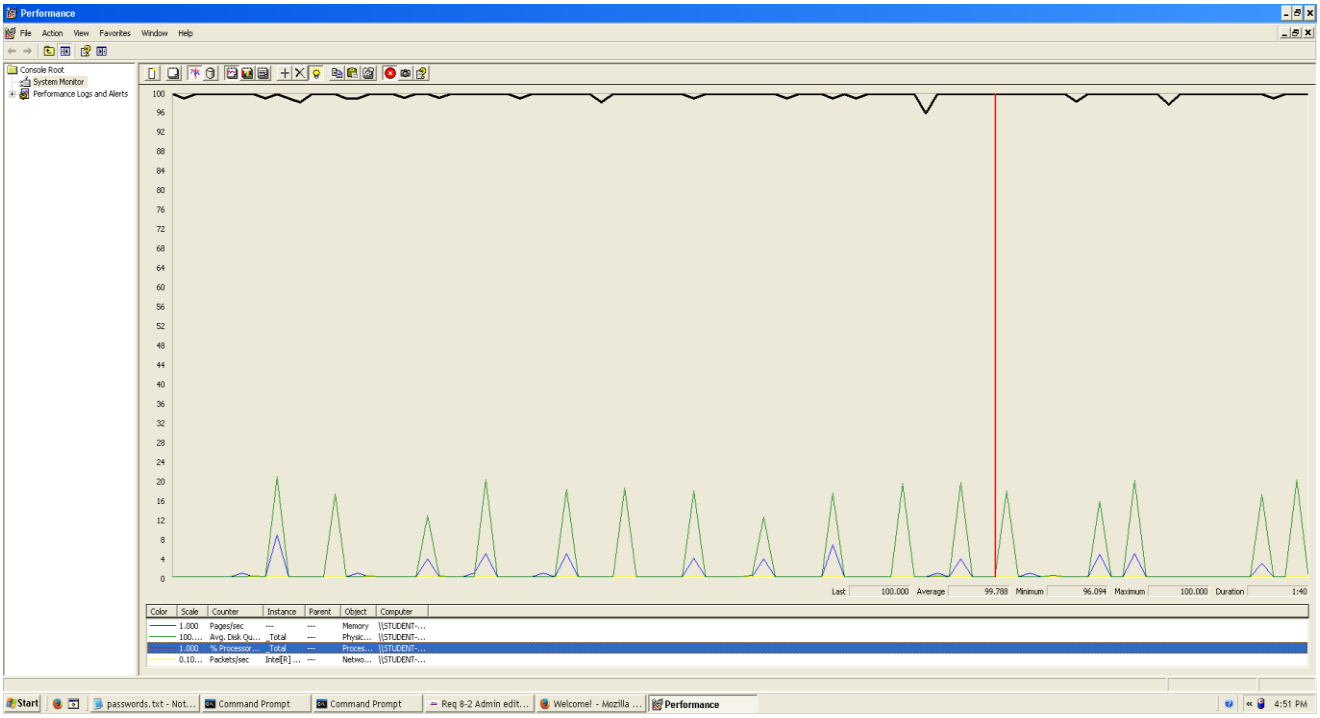
Overload test case: 365 concurrent users and 20 of loop count:

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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	21900	318	140	808	0	10440	0.06%	78.6/sec	298.5
/security/login.do	7300	300	125	749	2	10621	0.00%	27.0/sec	103.5
/j_spring_security_check	7300	550	318	1303	5	11028	0.00%	27.0/sec	116.6
/actor/display.do	14600	352	159	889	4	7917	0.00%	53.5/sec	296.7
/actor/administrator,co...	14600	605	309	1513	4	9532	0.00%	53.8/sec	353.9
/j_spring_security_logout	7300	370	174	938	2	6539	0.00%	27.1/sec	104.0
TOTAL	73000	409	186	1036	0	11028	0.02%	261.9/sec	1247.6



As we can see in the aggregate report, there's not so many errors, the maximum was 0.06% in the welcome view. 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.



Conclusion: The maximum number of concurrent users supported by this test case is 355 and we could improve it by assigning more CPU's resources to our system.

Req. 8.2 – An actor who is authenticated must be able to: Edit his or her personal data (company).

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

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

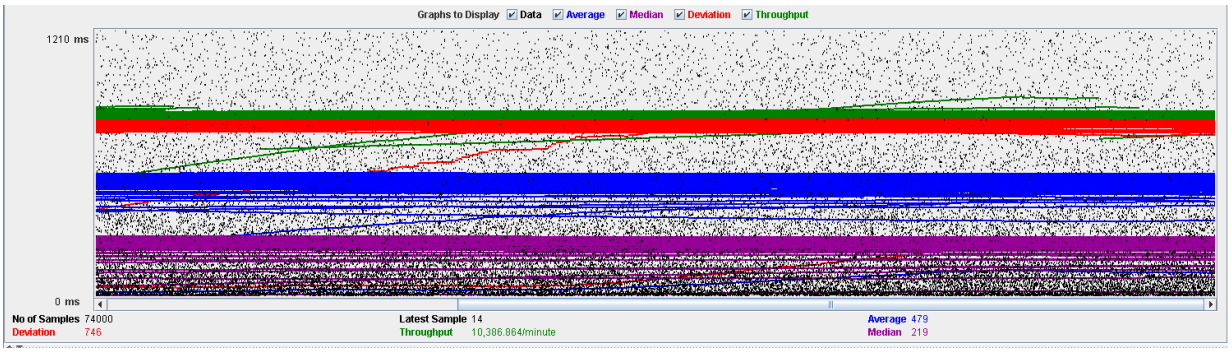
Test case description:

- The user log in the system.
- The user goes to his/her profile.
- The user goes to personal edit form.
- Change any field.
- Press save button save changes.
- Log out the system.

Maximum workload test case. 370 concurrent users and 20 of loop count:

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.

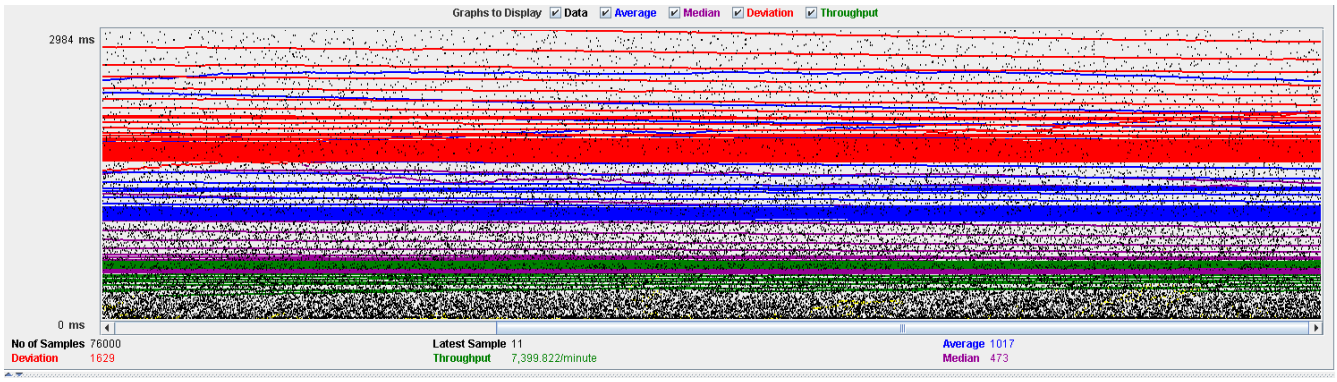
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	22200	357	171	934	3	6502	0.00%	51.9/sec	191.3
/security/login.do	7400	328	147	860	2	6330	0.00%	17.9/sec	68.5
/j_spring_security_check	7400	779	399	1929	6	13275	0.00%	17.9/sec	71.0
/actor/display.do	14800	382	190	966	4	6498	0.00%	35.3/sec	192.8
/actor/administrator,com...	14800	718	343	1838	5	12934	0.00%	35.5/sec	230.1
/j_spring_security_logout	7400	416	198	1067	4	6897	0.00%	17.9/sec	68.8
TOTAL	74000	479	219	1192	2	13275	0.00%	173.1/sec	806.2



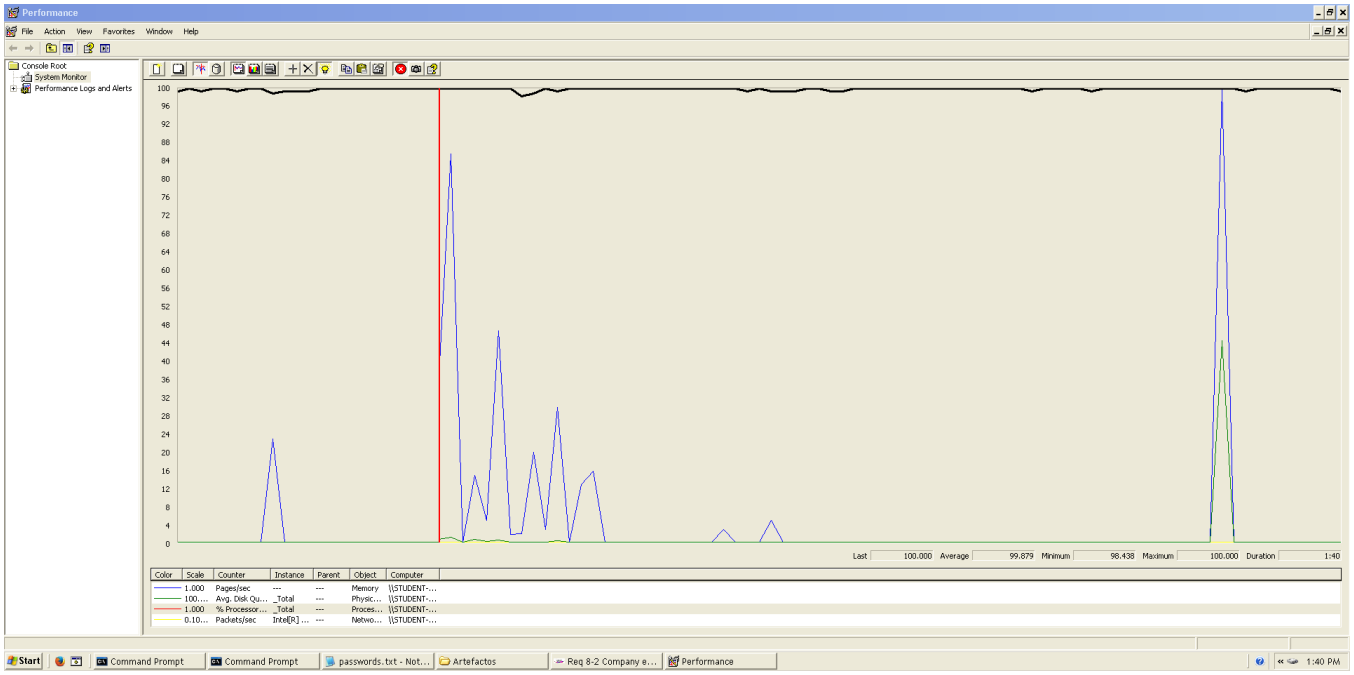
Overload test case: 380 concurrent users and 20 of loop count:

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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	22800	786	372	1959	1	26229	2.93%	37.0/sec	134.3
/security/login.do	7600	810	392	1989	1	22796	2.57%	12.7/sec	48.1
/j_spring_security_check	7600	1947	1061	4781	2	41799	3.36%	12.7/sec	49.4
/actor/display.do	15200	795	392	1941	2	21953	2.75%	25.3/sec	134.4
/actor/administrator,com...	15200	1337	645	3541	2	27767	2.99%	25.4/sec	159.7
/j_spring_security_logout	7600	793	379	1940	1	16221	2.89%	12.9/sec	48.7
TOTAL	76000	1017	473	2548	1	41799	2.91%	123.3/sec	562.0



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.



Conclusion: The maximum number of concurrent users supported by this test case is 370 and we could improve it by assigning more CPU's resources to our system.

Req. 8.2 – An actor who is authenticated must be able to: Edit his or her personal data (hacker).

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

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

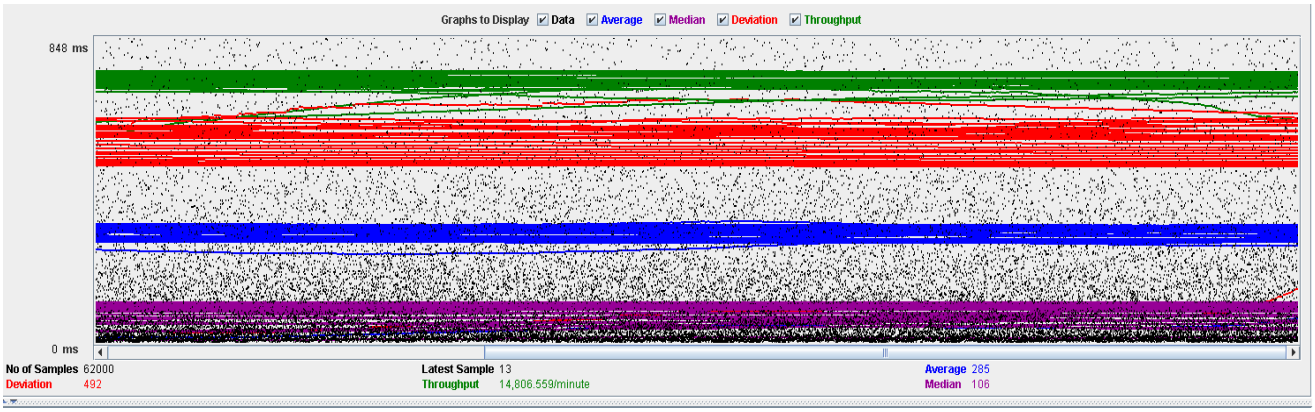
Test case description:

- The user log in the system.
- The user goes to his/her profile.
- The user goes to personal edit form.
- Change any field.
- Press save button save changes.
- Log out the system.

Maximum workload test case. 310 concurrent users and 20 of loop count:

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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	18600	220	77	589	2	6097	0.00%	74.0/sec	279.1
/security/login.do	6200	181	60	477	3	5397	0.00%	25.6/sec	97.8
/j_spring_security_check	6200	292	134	723	5	7925	0.00%	25.6/sec	108.1
/actor/display.do	12400	254	93	665	4	6805	0.00%	50.5/sec	276.7
/actor/administrator,com...	12400	467	194	1247	5	7750	0.00%	50.8/sec	331.1
/j_spring_security_logout	6200	280	120	729	3	5380	0.00%	25.6/sec	98.2
TOTAL	62000	285	106	745	2	7925	0.00%	246.8/sec	1164.2



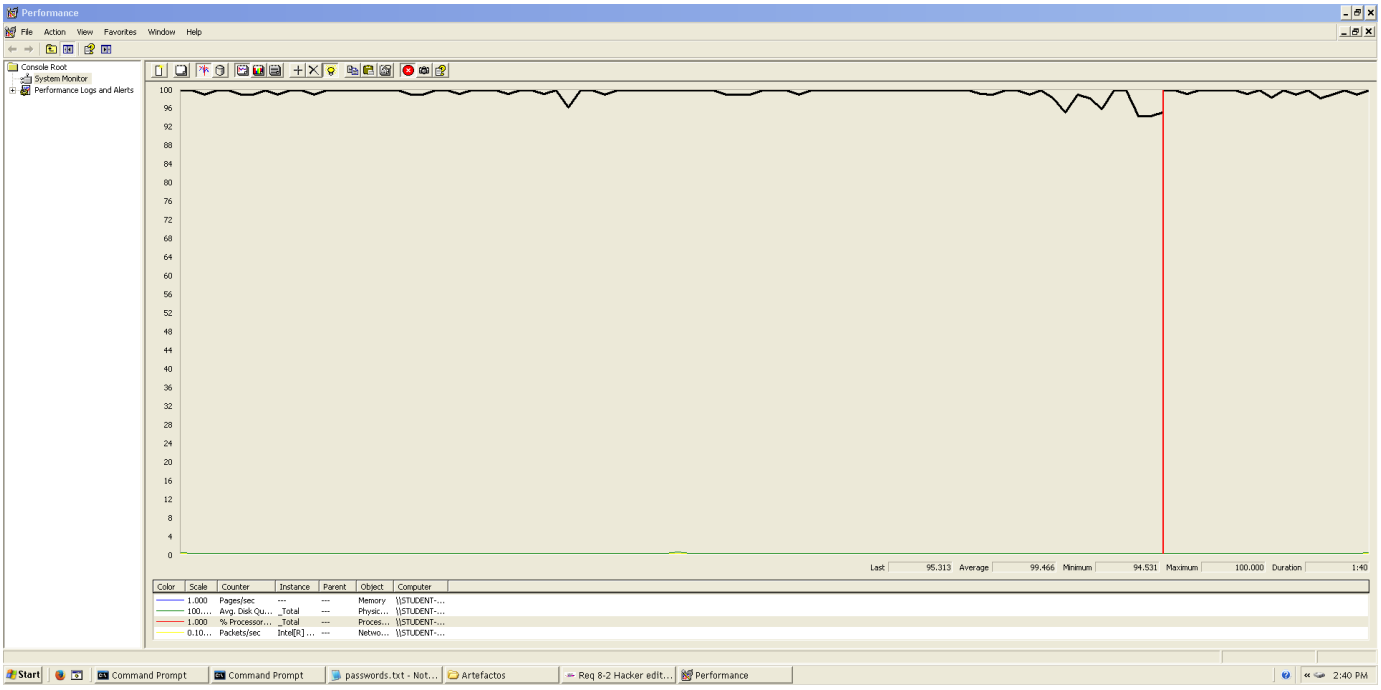
Overload test case: 320 concurrent users and 20 of loop count:

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.BindException) caught when connecting to the target host:

Address already in use: connect 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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	19200	235	71	636	1	6837	6.73%	69.2/sec	251.6
/security/login.do	6400	222	58	612	1	5278	7.14%	23.8/sec	87.7
/j_spring_security_check	6400	404	167	1082	1	8233	6.83%	23.8/sec	96.8
/actor/display.do	12800	283	93	773	2	6792	7.07%	47.2/sec	243.7
/actor/administrator,com...	12800	488	173	1396	2	7289	7.24%	47.5/sec	289.6
/j_spring_security_logout	6400	324	93	932	2	7267	7.36%	23.9/sec	87.9
TOTAL	64000	320	100	885	1	8233	7.02%	230.7/sec	1034.8

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 but not improve it so much because we can see it doesn't work always at 100%.



Conclusion: The maximum number of concurrent users supported by this test case is 310 and we could improve it by assigning more CPU's resources to our system.

Req. 9.1 – An actor who is authenticated as company must be able to creating and deleting their positions.

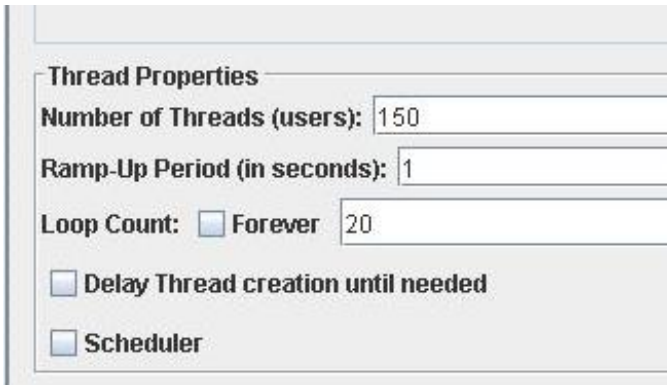
Technical details of the computer on which the test has been executed:

- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

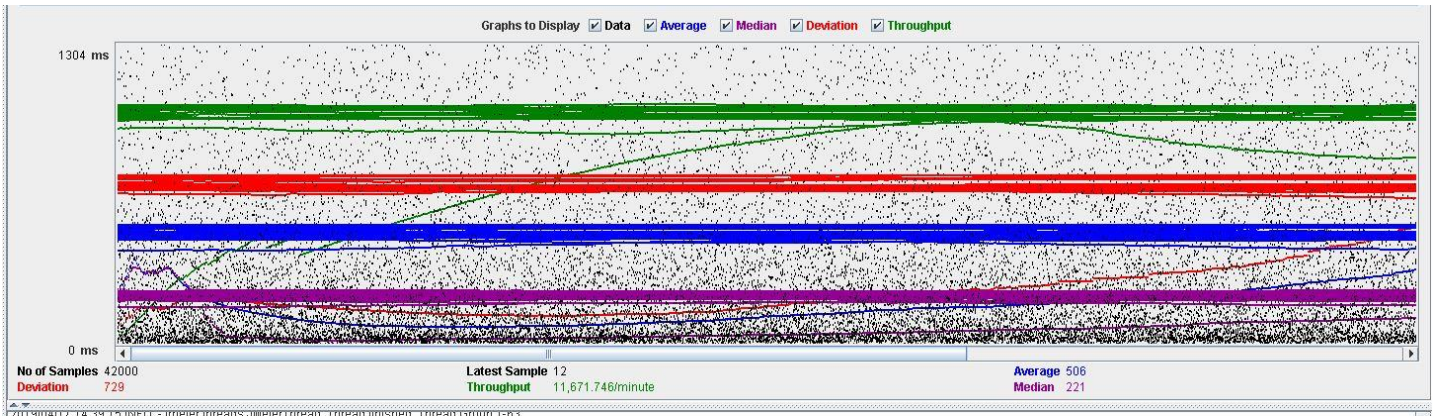
Test case description:

We initiate session in the system and we go to the page of the data of the actor. We click on position and it takes us to the list of positions, we create a new position. Once created we give it to edit it and we delete it. We relog the system.

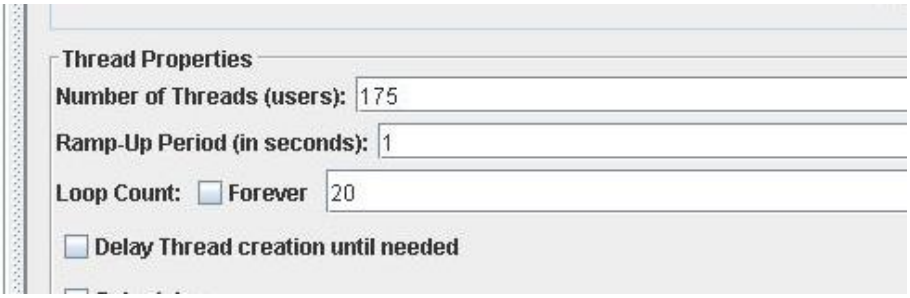
Maximum workload test case. 150 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	9000	399	157	1056	0	8508	0.00%	41.7/sec	152.9
/security/login.do	3000	506	211	1438	0	5693	0.00%	14.1/sec	53.9
/j_spring_security_check	3000	950	641	2252	0	10320	0.00%	14.1/sec	55.6
/actor/display.do	3000	419	190	1129	0	6923	0.00%	14.1/sec	76.8
/position/list.do	9000	427	188	1165	0	5999	0.00%	42.0/sec	160.5
/position/company/creat...	3000	527	216	1429	0	7565	0.00%	14.1/sec	80.4
/position/company/edit.do	9000	605	293	1614	0	9185	0.00%	42.3/sec	221.0
/j_spring_security_logout	3000	386	150	988	0	5692	0.00%	14.2/sec	53.9
TOTAL	42000	506	221	1379	0	10320	0.00%	194.5/sec	844.8



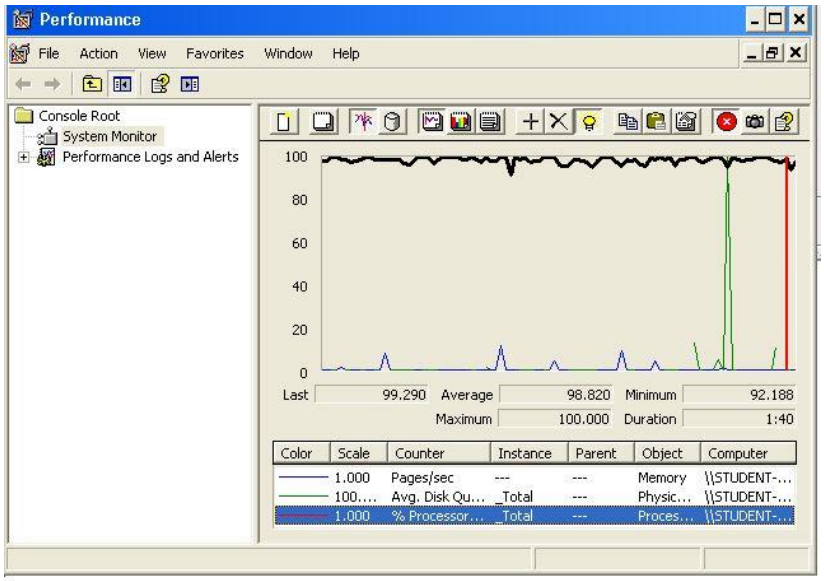
Overload test case: 175 concurrent users and 20 of loop count:



As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	10500	639	252	1745	0	11486	0.00%	35.4/sec	130.2
/security/login.do	3500	779	357	2123	0	17086	0.00%	11.9/sec	45.6
/j_spring_security_check	3500	1397	911	3260	0	14267	0.00%	11.9/sec	47.1
/actor/display.do	3500	655	289	1690	0	11317	0.00%	11.9/sec	64.9
/position/list.do	10500	657	282	1739	0	16147	0.00%	35.6/sec	136.1
/position/company/creat...	3500	806	372	2148	0	9243	0.00%	11.9/sec	68.0
/position/company/edit.do	10500	902	437	2354	0	16342	0.00%	35.7/sec	186.9
/j_spring_security_logout	3500	675	257	1783	0	13633	0.00%	12.0/sec	45.9
TOTAL	49000	779	345	2093	0	17086	0.00%	165.2/sec	719.2

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.



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. 9.1 – An actor who is authenticated as company must be able to displaying their positions.

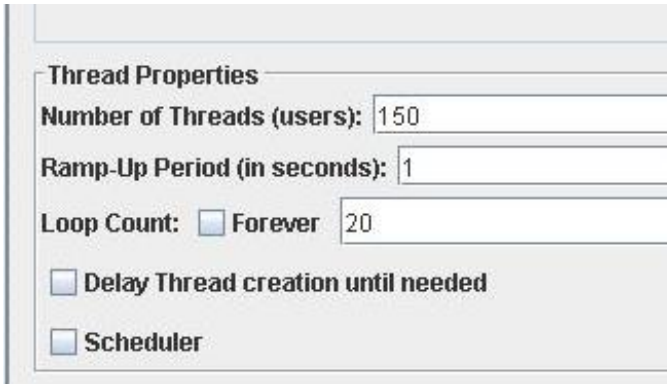
Technical details of the computer on which the test has been executed:

- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

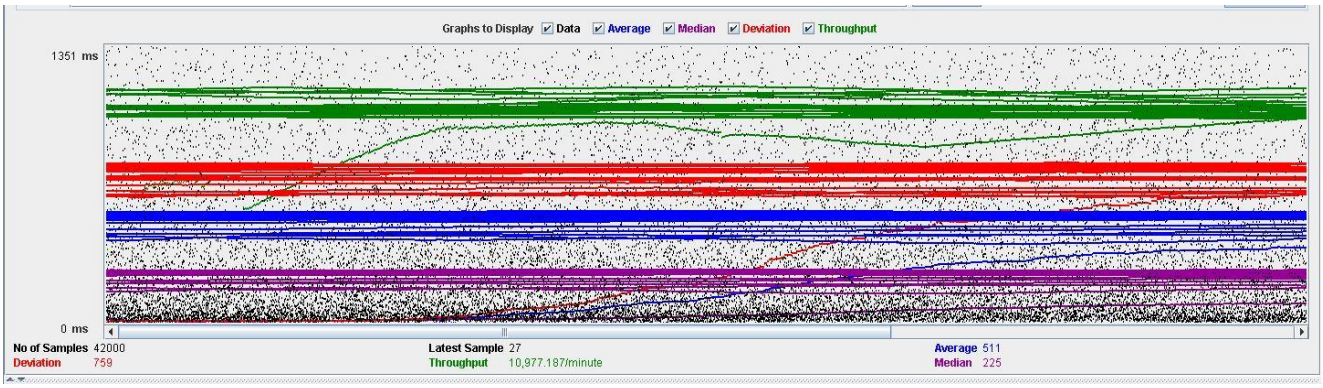
Test case description:

We initiate session in the system and we go to the page of the data of the actor. Click on position and take us to the list of positions, we show you a position. We relog the system.

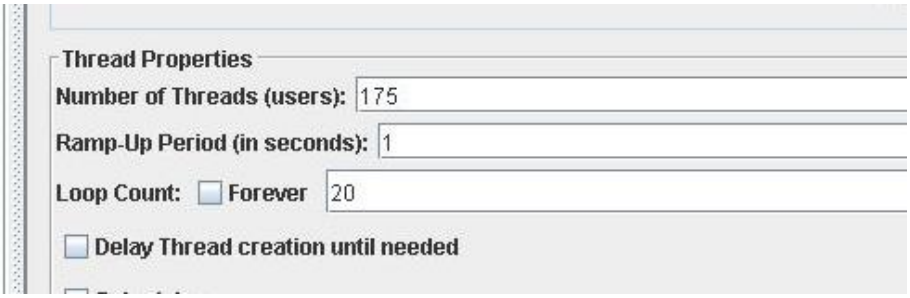
Maximum workload test case. 150 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	9000	393	149	1048	0	10200	0.00%	39.2/sec	143.8
/security/login.do	3000	555	236	1543	0	12046	0.00%	13.3/sec	50.5
/j_spring_security_check	3000	982	646	2332	0	9335	0.00%	13.3/sec	52.1
/actor/display.do	3000	429	193	1145	0	8445	0.00%	13.3/sec	72.0
/position/list.do	9000	431	184	1114	0	10143	0.00%	39.5/sec	150.6
/position/company/creat...	3000	499	221	1352	0	9098	0.00%	13.2/sec	75.5
/position/company/edit.do	9000	609	308	1548	0	13232	0.00%	39.7/sec	207.4
/j_spring_security_logout	3000	390	155	1027	0	8384	0.00%	13.4/sec	51.0
TOTAL	42000	511	225	1368	0	13232	0.00%	183.0/sec	794.6



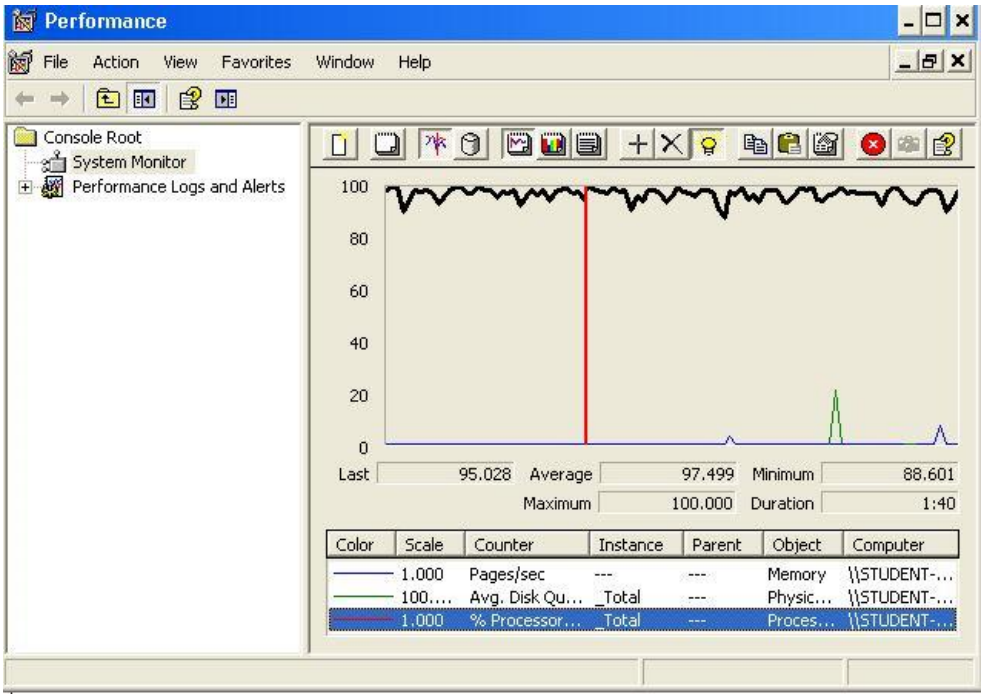
Overload test case: 175 concurrent users and 20 of loop count:



As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	10500	580	245	1577	0	10583	0.00%	36.6/sec	134.7
/security/login.do	3500	736	363	1959	0	13396	0.00%	12.4/sec	47.2
/j_spring_security_check	3500	1324	941	2992	0	13010	0.00%	12.4/sec	48.9
/actor/display.do	3500	615	270	1641	0	10167	0.00%	12.4/sec	67.3
/position/list.do	10500	641	291	1700	0	11753	0.00%	36.9/sec	141.0
/position/company/creat...	3500	762	400	2008	0	10282	0.00%	12.3/sec	70.5
/position/company/edit.do	10500	892	495	2277	0	15569	0.00%	37.0/sec	193.7
/j_spring_security_logout	3500	621	274	1639	0	10160	0.00%	12.4/sec	47.4
TOTAL	49000	743	356	1964	0	15569	0.00%	171.0/sec	744.4

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.



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. 9.1 – An actor who is authenticated as company must be able to editing their positions.

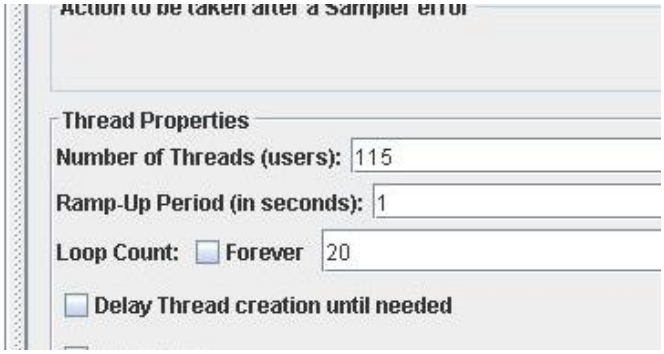
Technical details of the computer on which the test has been executed:

- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

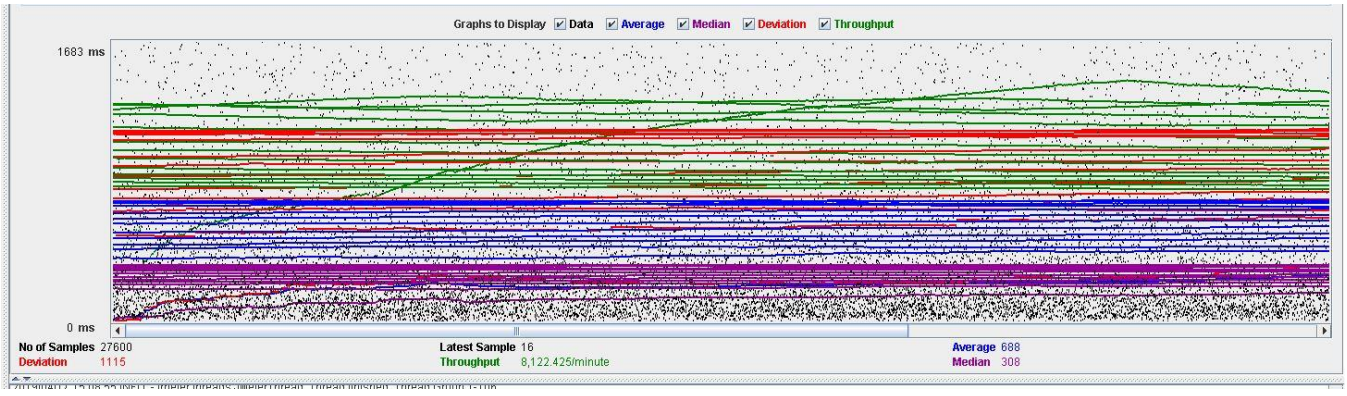
Test case description:

We initiate session in the system and we go to the page of the data of the actor. Click on position and take us to the list of positions, we give you to edit one. We relog the system.

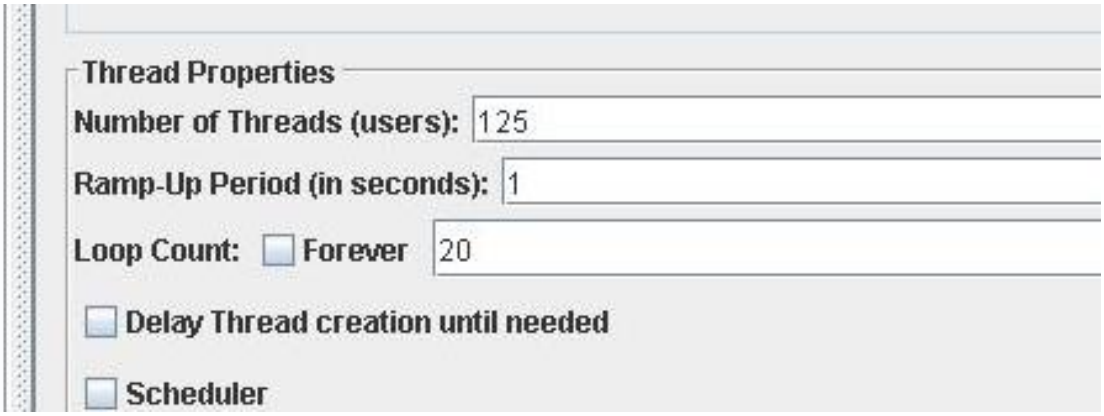
Maximum workload test case. 115 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	2300	543	217	1344	0	17036	0.00%	11.3/sec	43.1
/j_spring_security_check	2300	1067	620	2576	19	12665	0.00%	11.3/sec	44.4
/	4600	543	232	1313	0	19439	0.00%	22.6/sec	82.7
/problem/company/list.do	6900	566	243	1344	2	21561	0.00%	33.9/sec	213.4
/problem/company/creat...	2300	542	227	1346	0	12358	0.00%	11.3/sec	54.9
/problem/company/edit...	6900	927	481	2221	0	13335	0.00%	33.9/sec	176.8
/j_spring_security_logout	2300	538	226	1299	7	18869	0.00%	11.3/sec	43.2
TOTAL	27600	688	308	1675	0	21561	0.00%	135.4/sec	657.8

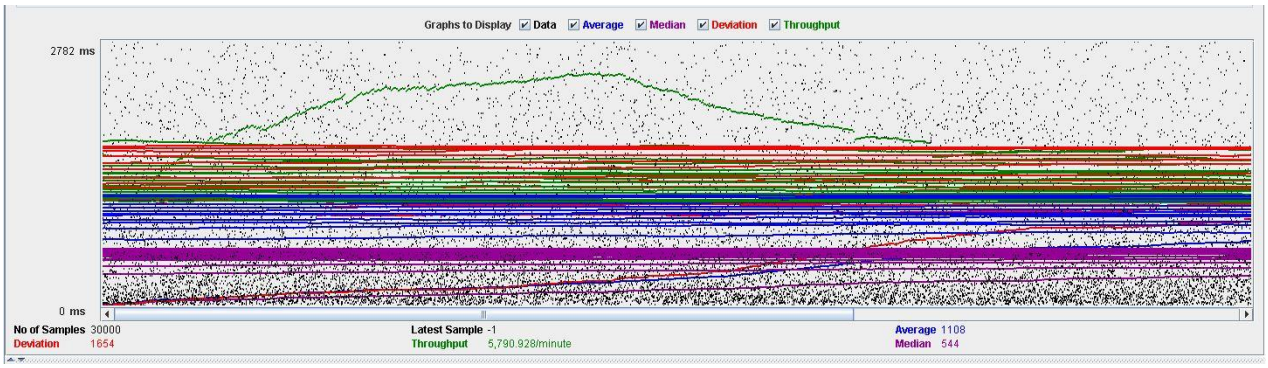


Overload test case: 125 concurrent users and 20 of loop count:

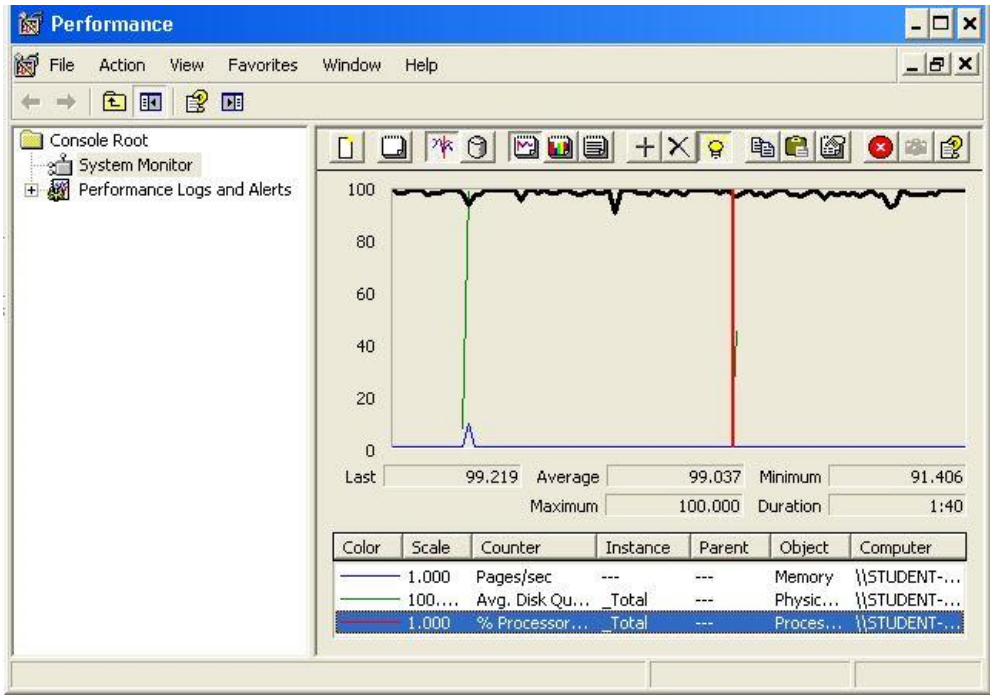


As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/securitylogin.do	2500	846	389	2071	0	21224	0.00%	8.1/sec	30.7
/j_spring_security_check	2500	1789	1088	4235	8	29925	0.00%	8.1/sec	31.7
/	5000	883	398	2236	0	26348	0.00%	16.1/sec	59.0
/problem/companylist.do	7500	899	436	2198	3	24083	0.00%	24.1/sec	152.3
/problem/companycreat...	2500	886	394	2245	0	25083	0.00%	8.1/sec	39.2
/problem/companyedit...	7500	1474	860	3499	0	33738	0.00%	24.2/sec	126.1
/j_spring_security_logout	2500	891	413	2255	10	16400	0.00%	8.1/sec	30.7
TOTAL	30000	1108	544	2779	0	33738	0.00%	96.5/sec	469.2



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.



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.

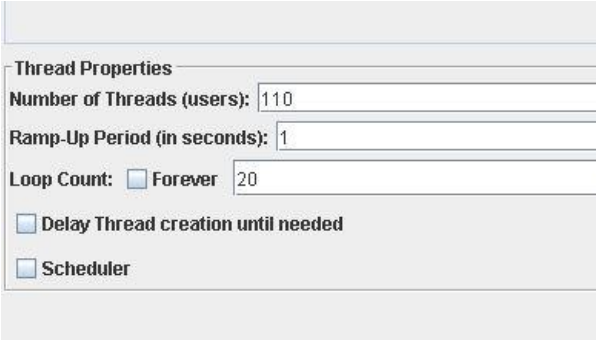
Req. 9.1 – An actor who is authenticated as company must be able to make final their positions in draft mode.

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

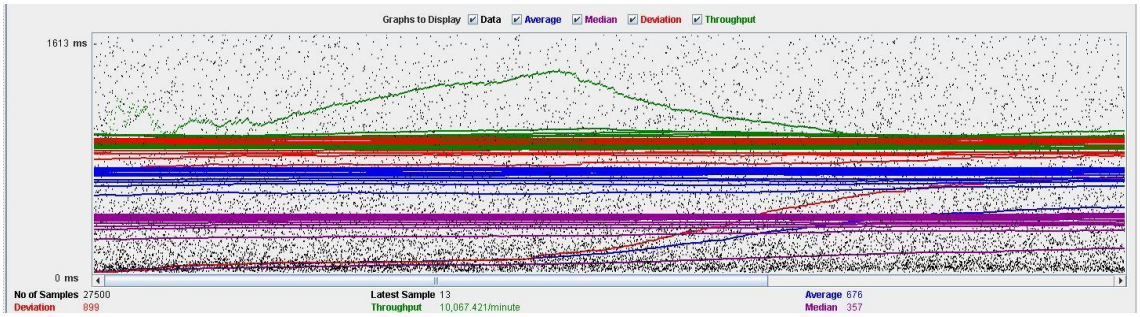
- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

Test case description:
We initiate session in the system and we go to the page of the data of the actor. We click on position and it takes us to the list of positions, we give it to show one and click on make final. We closed session in the system.

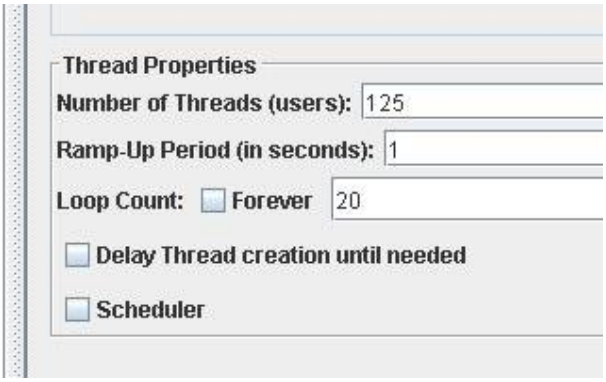
Maximum workload test case. concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	6600	472	209	1197	0	12457	0.00%	45.1/sec	165.5
/security/login.do	2200	495	203	1223	0	11099	0.00%	15.1/sec	57.5
/j_spring_security_check	2200	947	579	2241	16	9652	0.00%	15.1/sec	59.3
/actor/display.do	2200	485	209	1167	0	7943	0.00%	15.1/sec	81.8
/position/list.do	4400	507	231	1234	0	9124	0.00%	30.1/sec	159.1
/position/display.do	2200	500	230	1243	0	9946	0.00%	15.1/sec	76.1
/position/company/mak...	2200	1066	680	2339	52	11005	0.00%	15.1/sec	82.4
/j_spring_security_logout	2200	496	224	1256	10	6929	0.00%	15.1/sec	57.6
TOTAL	24200	582	271	1462	0	12457	0.00%	165.5/sec	737.8

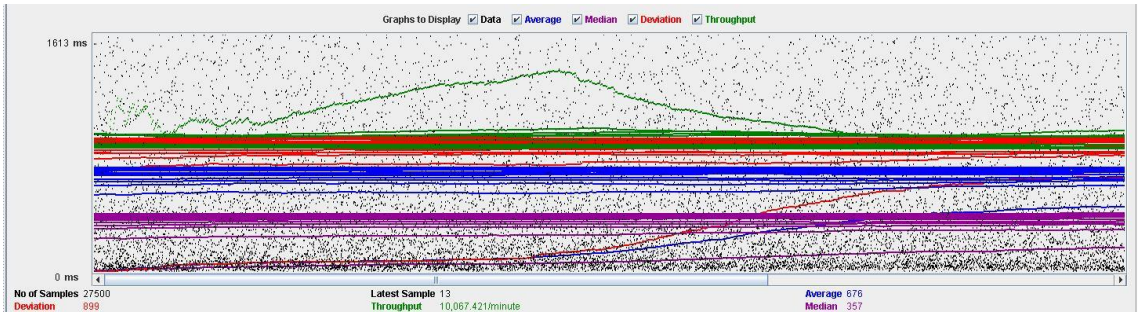


Overload test case: 125 concurrent users and 20 of loop count:

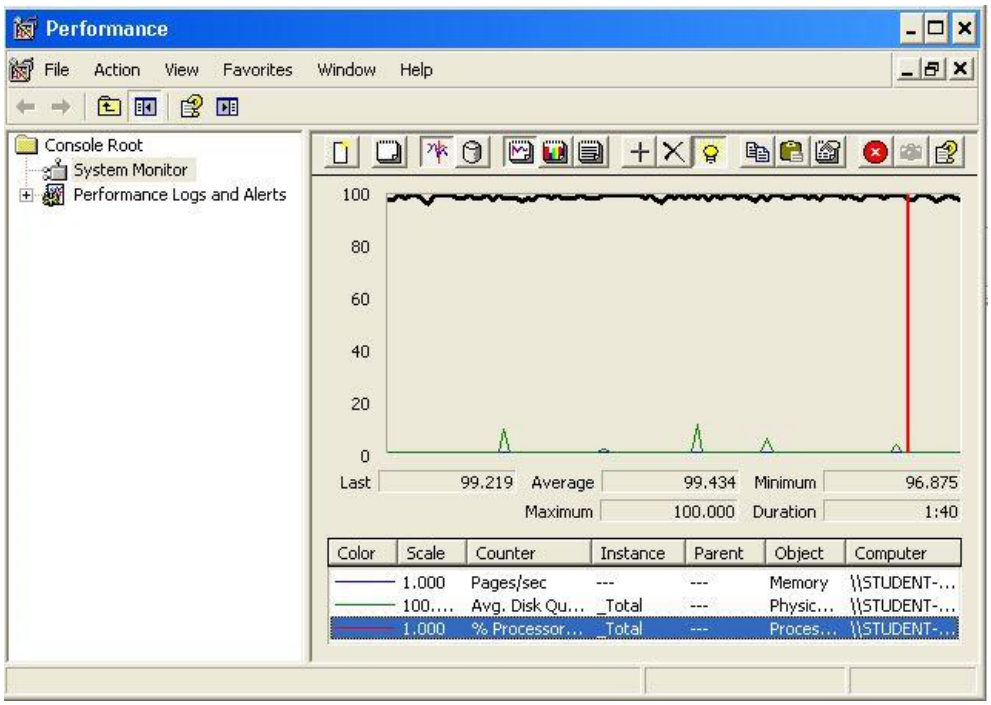


As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	7500	555	277	1442	0	9430	0.00%	45.8/sec	167.9
/securitylogin.do	2500	562	272	1487	0	7410	0.00%	15.3/sec	58.3
/j_spring_security_check	2500	1148	777	2597	10	12956	0.00%	15.3/sec	60.1
/actor/display.do	2500	551	275	1460	0	14536	0.00%	15.3/sec	82.9
/position/list.do	5000	575	296	1414	0	9542	0.00%	30.5/sec	161.2
/position/display.do	2500	573	294	1456	0	7270	0.00%	15.3/sec	77.1
/position/company/mak...	2500	1217	867	2672	51	17018	0.00%	15.3/sec	83.5
/j_spring_security_logout	2500	568	299	1440	8	9114	0.00%	15.3/sec	58.2
TOTAL	27500	676	357	1721	0	17018	0.00%	167.8/sec	748.3



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.



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.

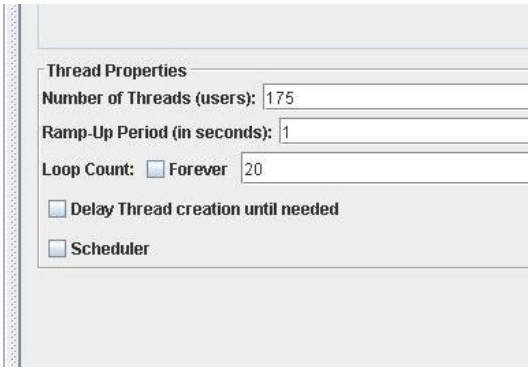
Req. 9.1 – An actor who is authenticated as company must be able to listing their positions.

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

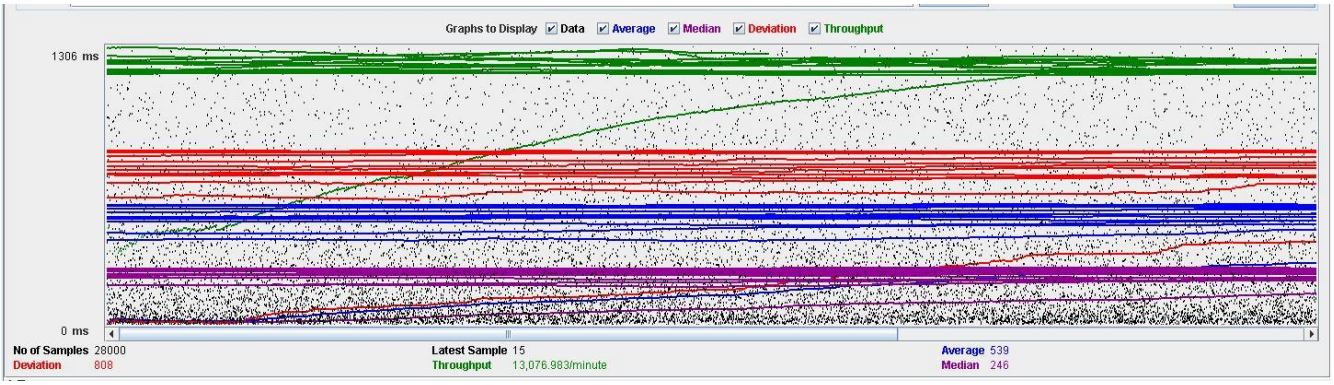
- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

Test case description:
We initiate session in the system and we go to the page of the data of the actor. Click on position and take us to the list of positions. We closed session in the system.

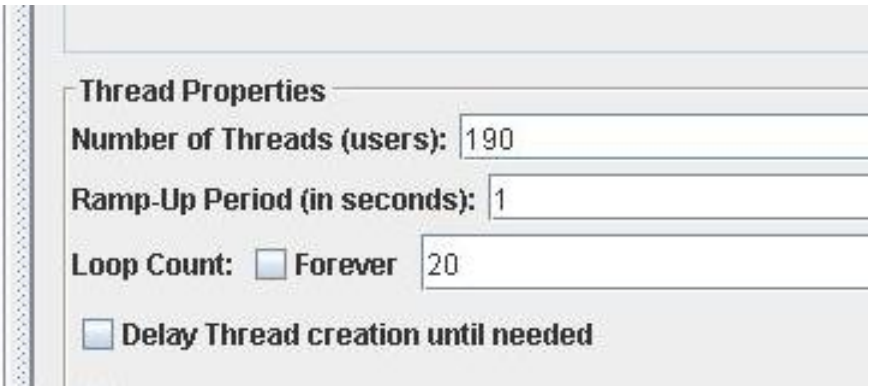
Maximum workload test case. 175 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	10500	454	195	1162	0	10143	0.00%	81.7/sec	300.5
/security/login.do	3500	512	229	1354	0	7594	0.00%	27.6/sec	105.4
/j_spring_security_check	3500	1008	678	2318	0	10668	0.00%	27.6/sec	109.0
/actor/display.do	3500	475	224	1195	0	7910	0.00%	27.6/sec	150.2
/position/list.do	3500	485	218	1248	0	9419	0.00%	27.6/sec	105.6
/j_spring_security_logout	3500	472	214	1225	0	8140	0.00%	27.6/sec	105.7
TOTAL	28000	539	246	1397	0	10668	0.00%	217.9/sec	868.9

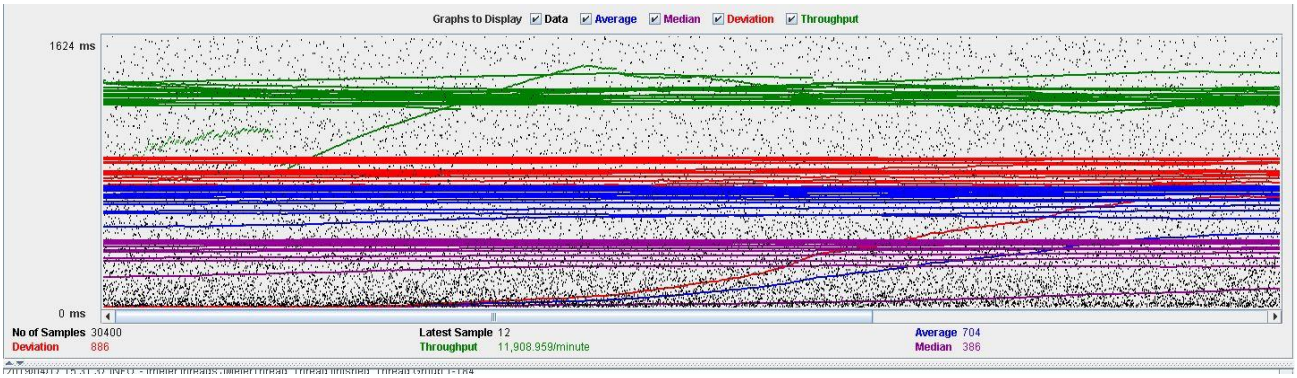


Overload test case: 190 concurrent users and 20 of loop count:

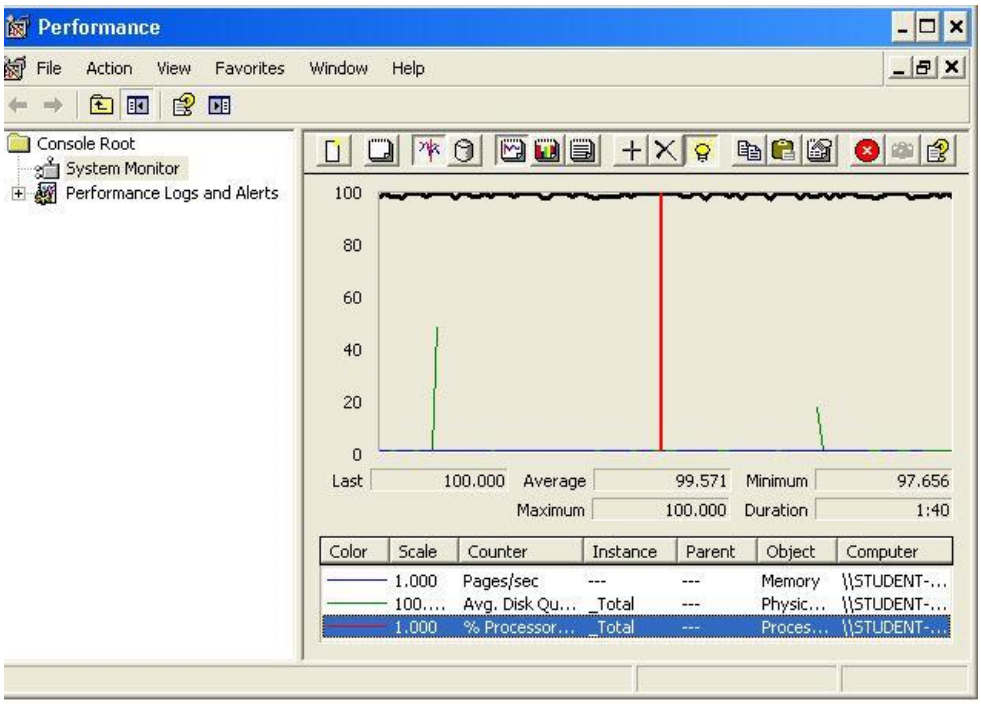


As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	11400	591	311	1539	0	12908	0.00%	74.4/sec	274.0
/security/login.do	3800	695	384	1762	0	7277	0.00%	25.0/sec	95.8
/j_spring_security_check	3800	1299	999	2769	0	12359	0.00%	25.0/sec	99.2
/actor/display.do	3800	625	338	1604	0	12946	0.00%	25.0/sec	136.4
/position/list.do	3800	634	341	1608	0	13643	0.00%	25.0/sec	95.9
/j_spring_security_logout	3800	606	316	1563	0	9549	0.00%	25.0/sec	96.0
TOTAL	30400	704	386	1806	0	13643	0.00%	198.5/sec	792.4



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.



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. 9.2 – An actor who is authenticated as a company must be able to creating and deleting their problems.

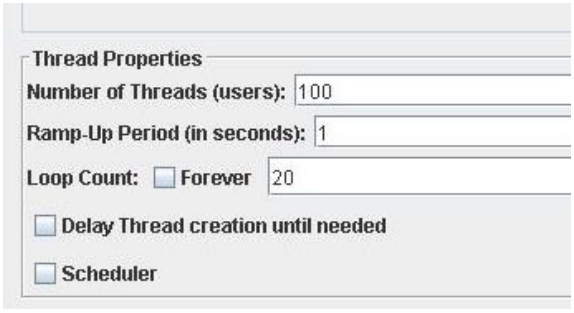
Technical details of the computer on which the test has been executed:

- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

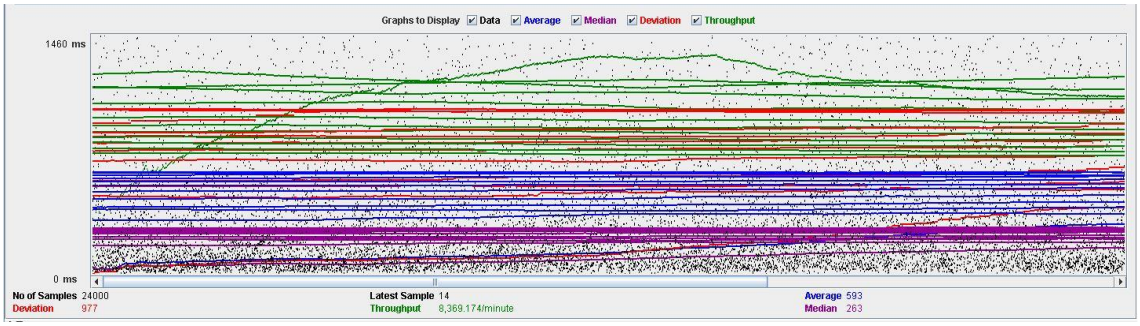
Test case description:

We initiate session in the system and we go to the page of the data of the actor. We click on position and it takes us to the list of problems, we create a new problem. Once created we give it to edit it and we delete it. We relog the system.

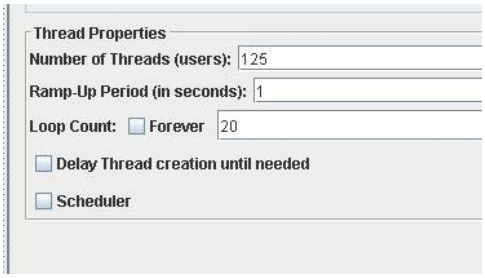
Maximum workload test case. 100 concurrent users and 20 of loop count:



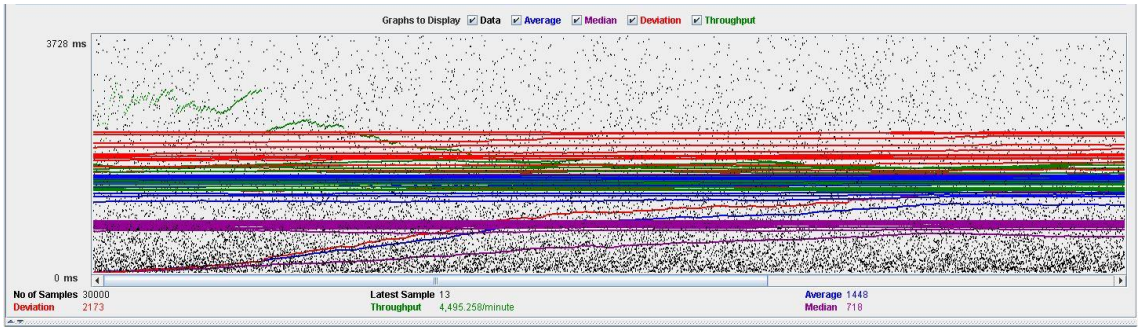
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	2000	432	170	1005	0	13786	0.00%	11.7/sec	45.3
/j_spring_security_check	2000	923	511	2250	6	11454	0.00%	11.7/sec	45.8
/	4000	464	188	1189	0	10926	0.00%	23.3/sec	85.2
/problem/company/list.do	6000	496	218	1198	3	21982	0.00%	34.9/sec	219.9
/problem/company/creat...	2000	440	189	1090	0	8938	0.00%	11.7/sec	56.7
/problem/company/edit...	6000	808	417	1988	0	32630	0.00%	34.9/sec	182.2
/j_spring_security_logout	2000	476	198	1155	5	26818	0.00%	11.7/sec	44.5
TOTAL	24000	593	263	1471	0	32630	0.00%	139.5/sec	678.5



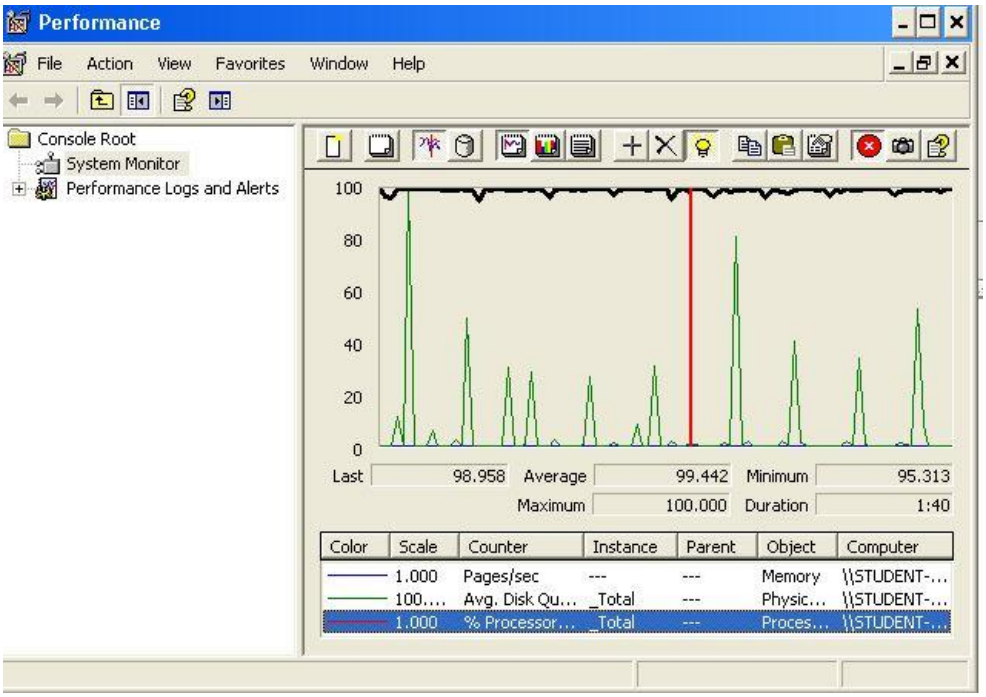
Overload test case: 125 concurrent users and 20 of loop count:



/security/login.do	2500	1149	497	2982	0	20904	0.00%	6.3/sec	24.3
/j_spring_security_check	2500	2287	1425	5152	9	51831	0.00%	6.3/sec	24.6
/	5000	1150	506	2852	0	31767	0.00%	12.5/sec	45.8
/problem/company/list.do	7500	1177	568	2819	15	34318	0.00%	18.7/sec	118.2
/problem/company/creat...	2500	1086	515	2662	0	19571	0.00%	6.2/sec	30.4
/problem/company/edit...	7500	1942	1141	4598	0	44148	0.00%	18.7/sec	97.8
/j_spring_security_logout	2500	1200	526	2979	5	31005	0.00%	6.3/sec	23.8
TOTAL	30000	1448	718	3607	0	51831	0.00%	74.9/sec	364.7



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.



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.

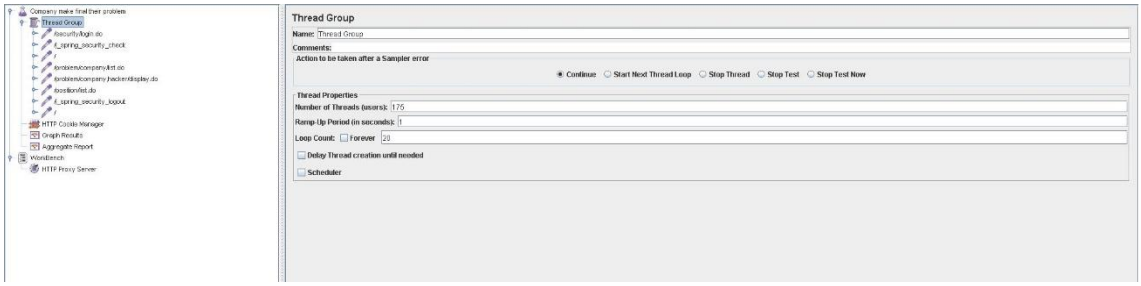
Req. 9.2 – An actor who is authenticated as a company must be able to listing and displaying their problems.

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

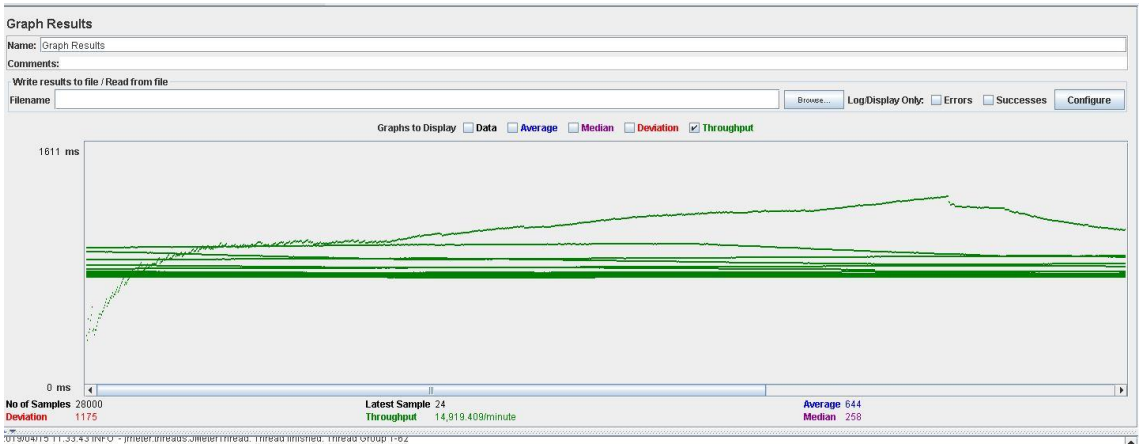
- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

Test case description:

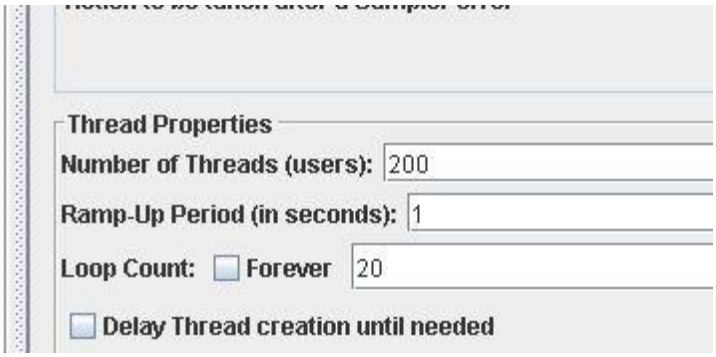
Maximum workload test case. 175 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	3500	528	207	1152	0	22407	0.00%	31.3/sec	122.2
/j_spring_security_check	3500	1057	561	2306	0	24007	0.00%	31.3/sec	124.1
/	7000	573	206	1277	0	20925	0.00%	62.6/sec	230.3
/problem/company/list.do	3500	552	216	1209	0	23028	0.00%	31.3/sec	163.6
/problem/company,hack...	3500	555	221	1256	0	15236	0.00%	31.3/sec	126.1
/position/list.do	3500	527	220	1170	0	21024	0.00%	31.3/sec	183.7
/j_spring_security_logout	3500	575	203	1220	0	26656	0.00%	31.3/sec	120.3
TOTAL	28000	617	243	1394	0	26656	0.00%	250.3/sec	1069.2

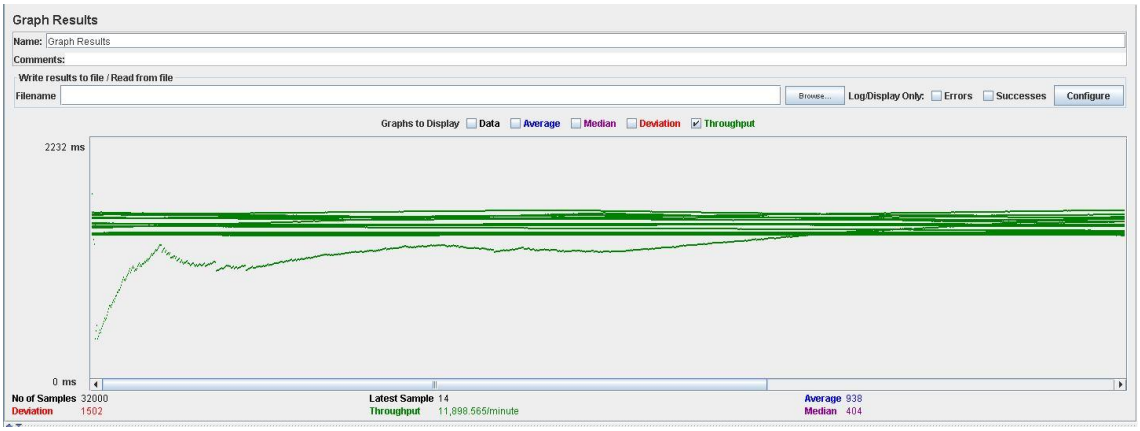


Overload test case:200 concurrent users and 20 of loop count:

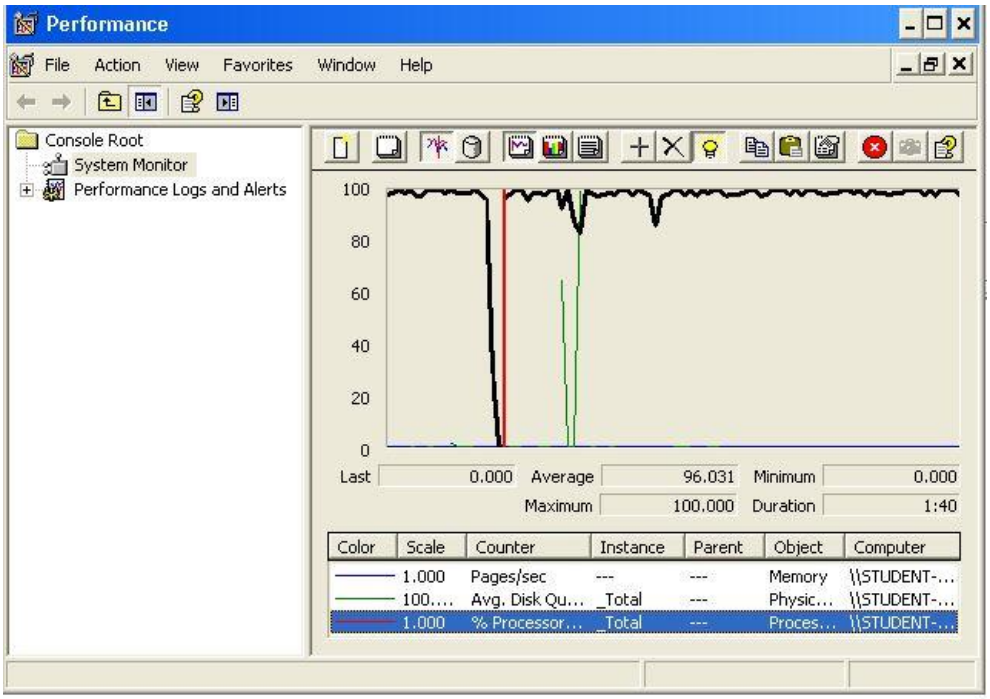


As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	3500	576	216	1363	0	15044	0.00%	31.1/sec	119.3
/j_spring_security_check	3500	1137	643	2697	0	29734	0.00%	31.1/sec	123.4
/	7000	572	223	1397	0	18870	0.00%	62.2/sec	228.8
/problem/company/list.do	3500	534	212	1336	0	17578	0.00%	31.1/sec	125.9
/problem/company,hack...	3500	581	229	1453	0	17050	0.00%	31.1/sec	124.1
/position/list.do	3500	604	245	1466	0	15992	0.00%	31.1/sec	119.3
/j_spring_security_logout	3500	578	228	1365	0	22556	0.00%	31.1/sec	119.6
TOTAL	28000	644	258	1593	0	29734	0.00%	248.7/sec	959.1



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.



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. 9.2 – An actor who is authenticated as a company must be able to make final their problems.

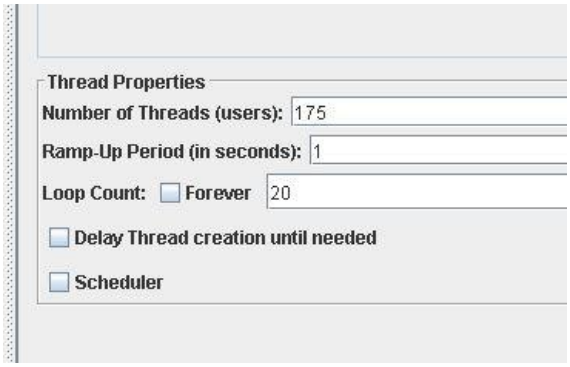
Technical details of the computer on which the test has been executed:

- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

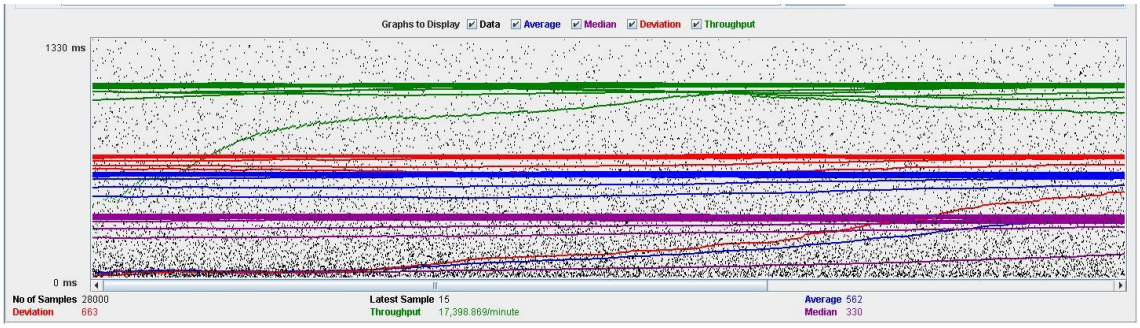
Test case description:

We initiate session in the system. We click on problems and it takes us to the list of problems, we give it to show one and click on make final. We closed session in the system.

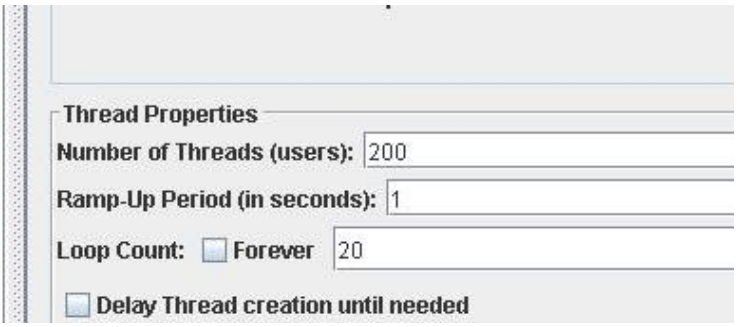
Maximum workload test case. 175 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	3500	430	234	1109	0	3935	0.00%	36.3/sec	141.6
/j_spring_security_check	3500	901	658	2009	1	7040	0.00%	36.3/sec	144.0
/	7000	452	250	1160	0	4910	0.00%	72.6/sec	267.1
/problem/company/list.do	7000	448	239	1149	0	4575	0.00%	72.6/sec	293.0
/problem/company/mak...	3500	896	662	2021	2	5169	0.00%	36.3/sec	153.6
/j_spring_security_logout	3500	468	258	1213	0	5129	0.00%	36.3/sec	139.7
TOTAL	28000	562	330	1423	0	7040	0.00%	290.0/sec	1137.0

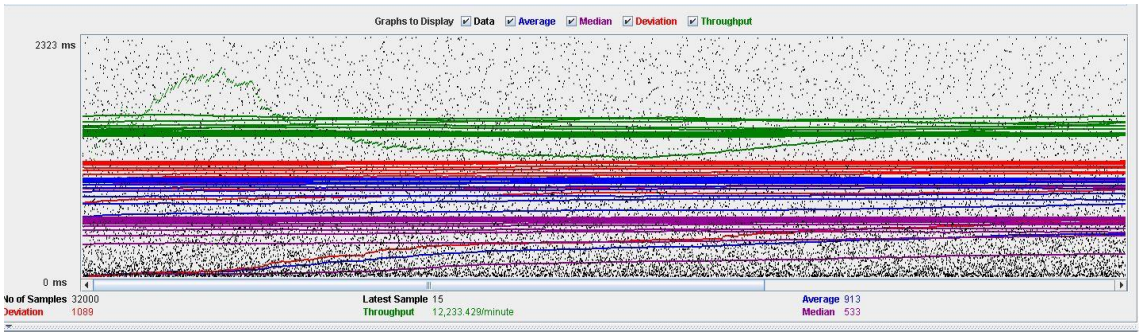


Overload test case:200 concurrent users and 20 of loop count:

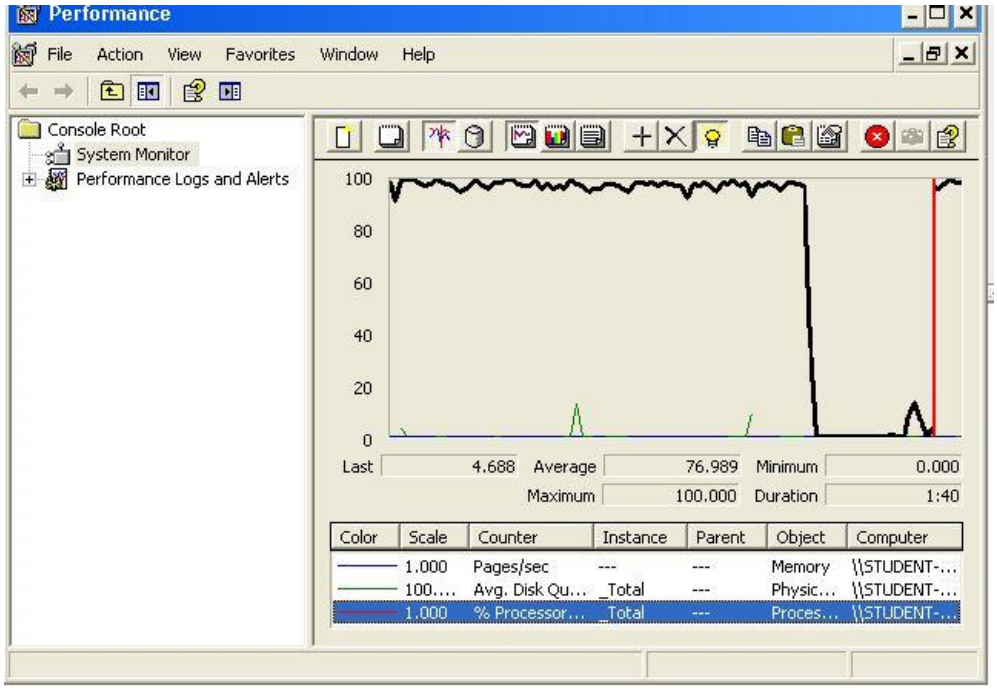


As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	4000	739	390	1948	0	11049	0.00%	25.5/sec	99.5
/j_spring_security_check	4000	1445	1094	3181	9	11280	0.00%	25.5/sec	101.1
/	8000	741	400	1930	0	9646	0.00%	51.0/sec	187.7
/problem/company/list.do	8000	712	395	1829	0	10252	0.00%	51.0/sec	205.8
/problem/company/mak...	4000	1475	1098	3192	13	12028	0.00%	25.5/sec	107.9
/j_spring_security_logout	4000	740	417	1877	0	11145	0.00%	25.5/sec	98.1
TOTAL	32000	913	533	2309	0	12028	0.00%	203.9/sec	799.6



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.



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. 9.1 – An actor who is authenticated as a company must be able to editing their problems.

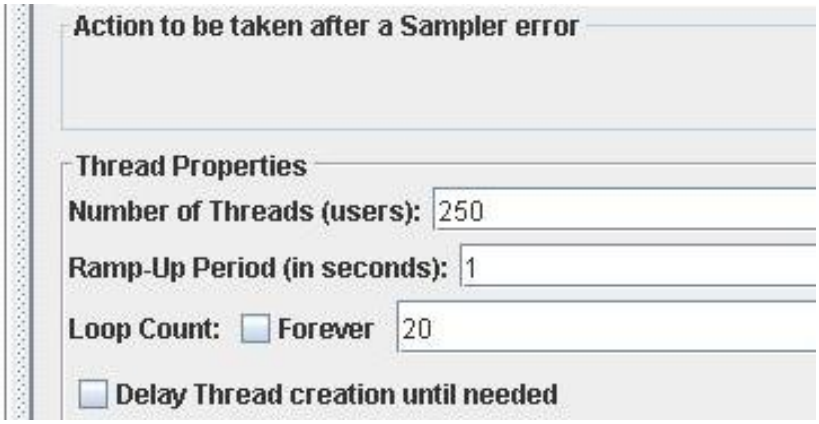
Technical details of the computer on which the test has been executed:

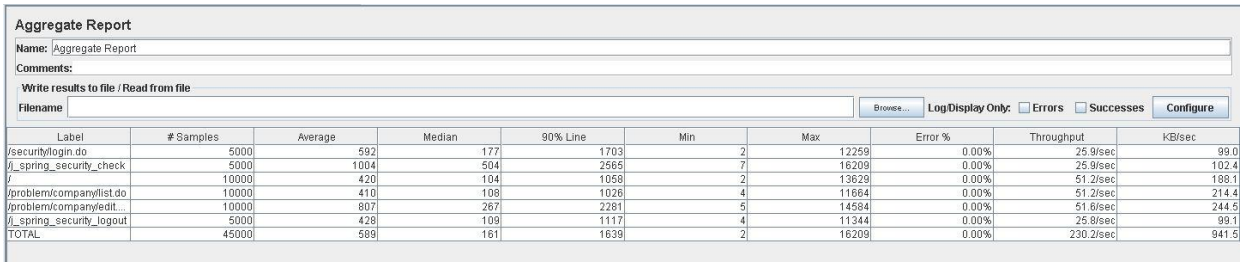
- Ram: 8 GB
- CPU: Intel core i7-8550U 1,8GHz
- Hard disk: 1TB
- Network card: Intel(R) Dual Band Wireless-AC 3165

Test case description:

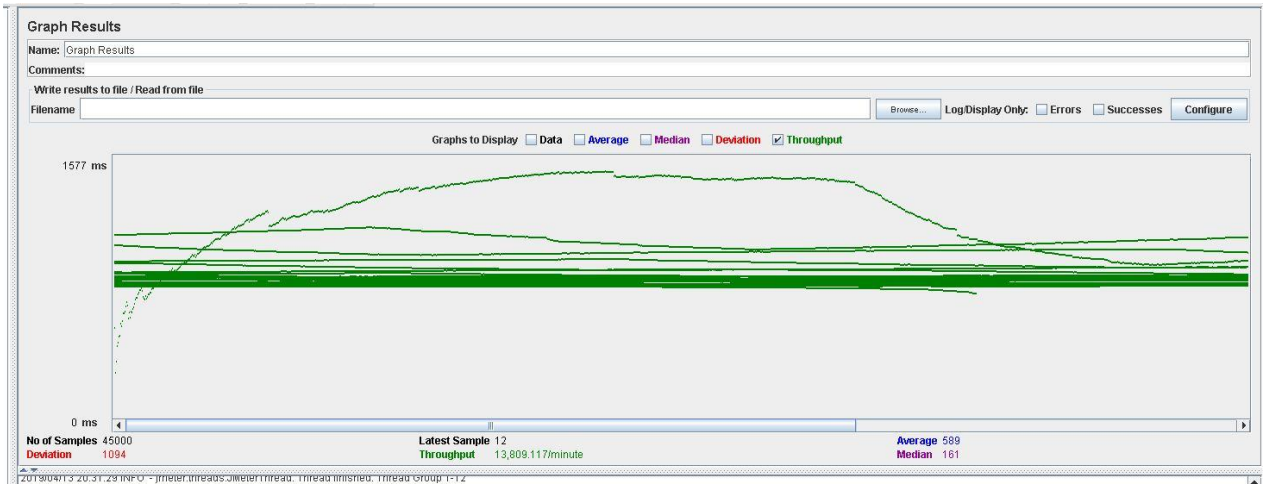
We initiate session in the system and we go to problems. Click on position and take us to the list of problems, we give you to edit one. We relog the system.

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

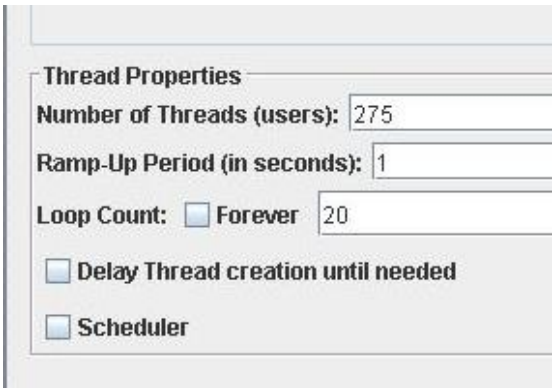




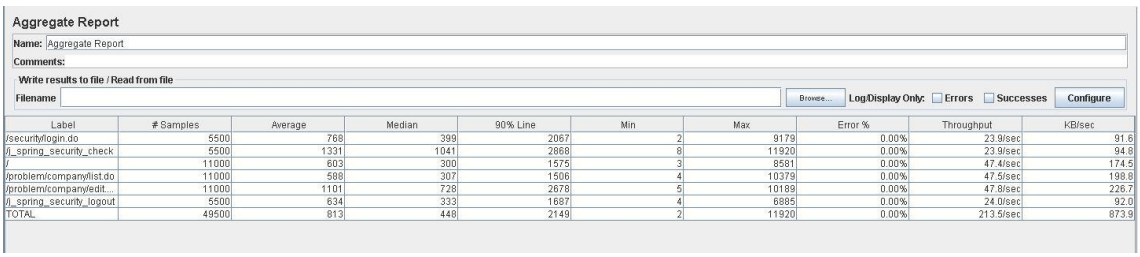
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	5000	592	177	1703	2	12259	0.00%	25.9/sec	99.0
/j_spring_security_check	5000	1004	504	2565	7	16209	0.00%	25.9/sec	102.4
/	10000	420	104	1058	2	13629	0.00%	51.2/sec	188.1
/problem/company/list.do	10000	410	108	1026	4	11664	0.00%	51.2/sec	214.4
/problem/company/edit...	10000	807	267	2281	5	14584	0.00%	51.6/sec	244.5
/j_spring_security_logout	5000	428	109	1117	4	11344	0.00%	25.8/sec	99.1
TOTAL	45000	589	161	1639	2	16209	0.00%	230.2/sec	941.5



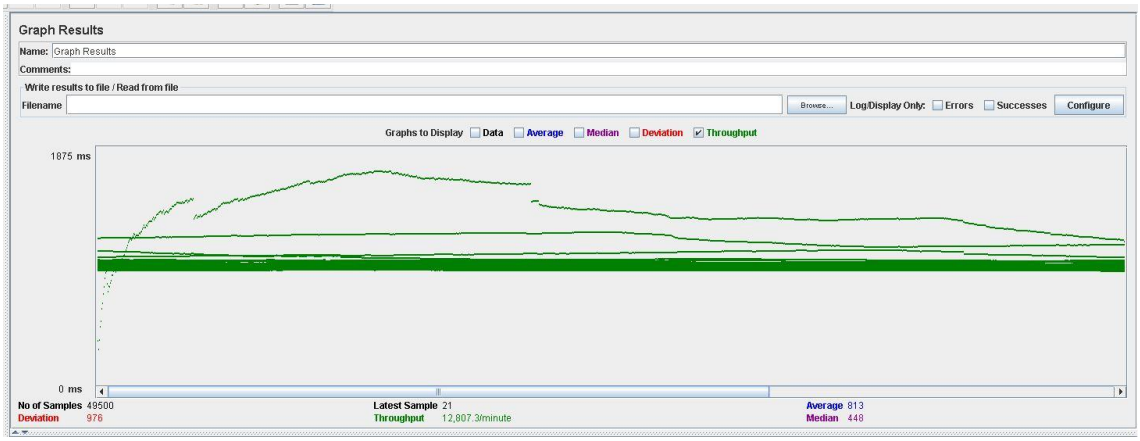
Overload test case:275 concurrent users and 20 of loop count:



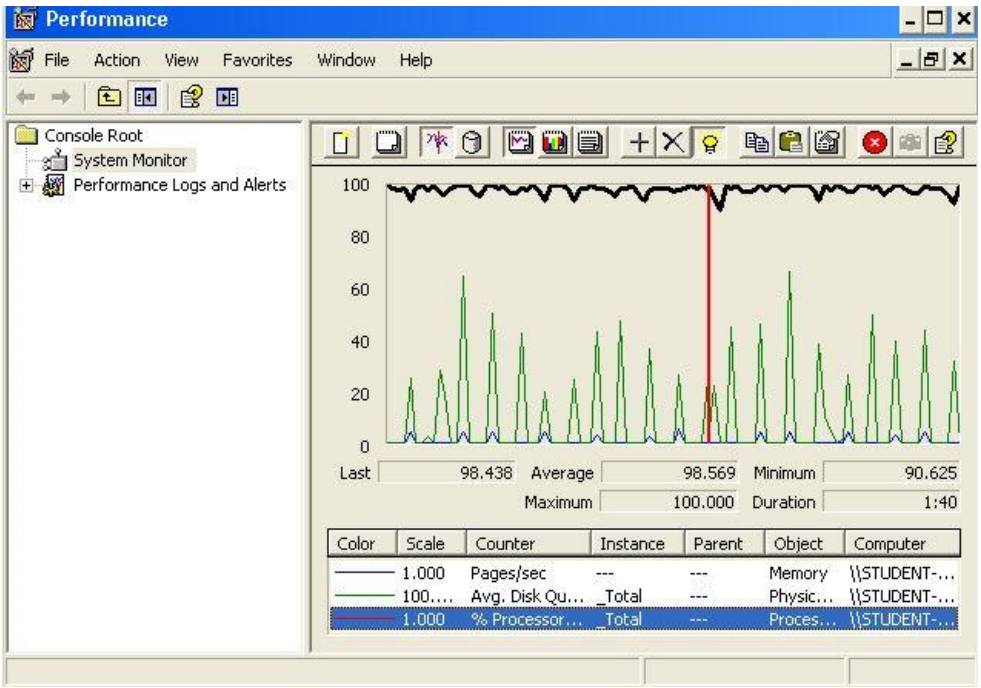
As we can see in the report if we increase the number of threads the 90% line gets a value too high.



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	5500	768	399	2067	2	9179	0.00%	23.9/sec	91.6
/j_spring_security_check	5500	1331	1041	2868	8	11920	0.00%	23.9/sec	94.8
/	11000	603	300	1575	3	8581	0.00%	47.4/sec	174.5
/problem/company/list.do	11000	588	307	1506	4	10379	0.00%	47.5/sec	198.8
/problem/company/edit...	11000	1101	728	2678	5	10189	0.00%	47.8/sec	226.7
/j_spring_security_logout	5500	634	333	1687	4	6885	0.00%	24.0/sec	92.0
TOTAL	49500	813	448	2149	2	11920	0.00%	213.5/sec	873.9



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.



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. 9.3 – Manage the applications to their positions, which includes listing them grouped by status, showing them, and updating them. Updating an application amounts to making a decision on it: an application whose status is SUBMITTED may change to status ACCEPTED or REJECTED.

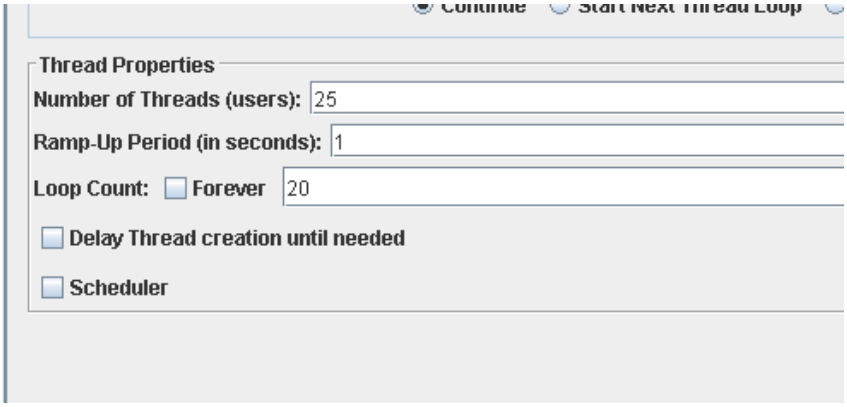
Technical details of the computer on which the test has been executed:

- Ram: 8,0 (1x) GB, DDR3 RAM (1,600 MHz)
- CPU: Intel Core i5-4200U
- Disco duro: 240 GB SSD

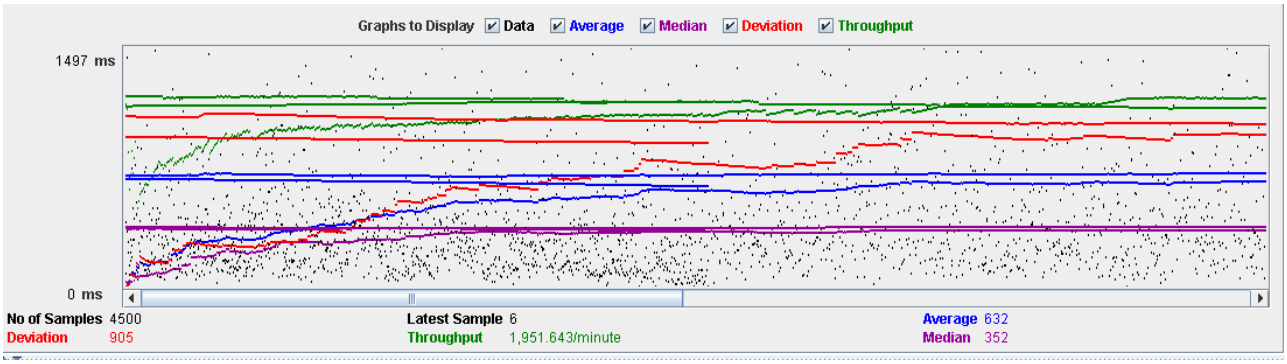
Test case description:

- The user logs in as a company.
- Display her/his profile.
- List positions.
- Display position.
- List applications.
- Rejected application.
- The company closes session.

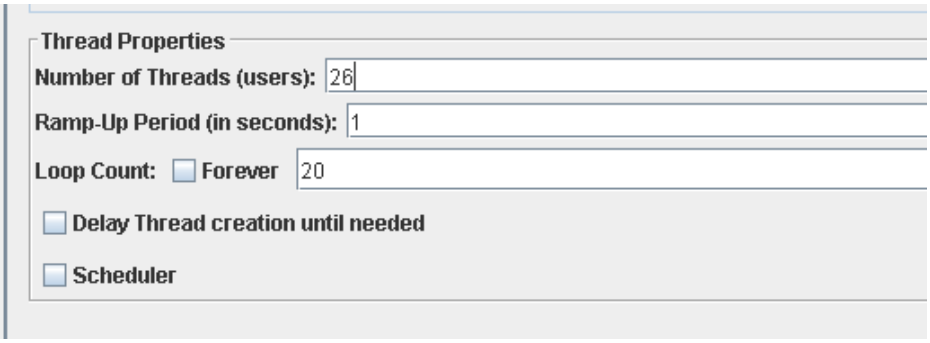
Maximum workload test case. 25 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	480	421	225	1001	7	5004	0.00%	3.1/sec	11.9
/j_spring_security_check	480	940	605	2126	43	7470	0.00%	3.1/sec	12.3
/actor/display.do	480	530	315	1172	4	5598	0.00%	3.1/sec	17.0
/position/list.do	480	541	361	1081	26	8660	0.00%	3.1/sec	17.3
/position/display.do	480	619	418	1398	15	4617	0.00%	3.2/sec	15.9
/application/company/list.do	960	668	457	1276	56	10713	0.00%	6.3/sec	30.1
/application/company/accept.do	480	1075	765	2253	64	11547	0.00%	3.2/sec	12.3
/j_spring_security_logout	480	479	265	1080	5	5746	0.00%	3.2/sec	12.1
TOTAL	4320	660	418	1435	4	11547	0.00%	27.9/sec	127.2



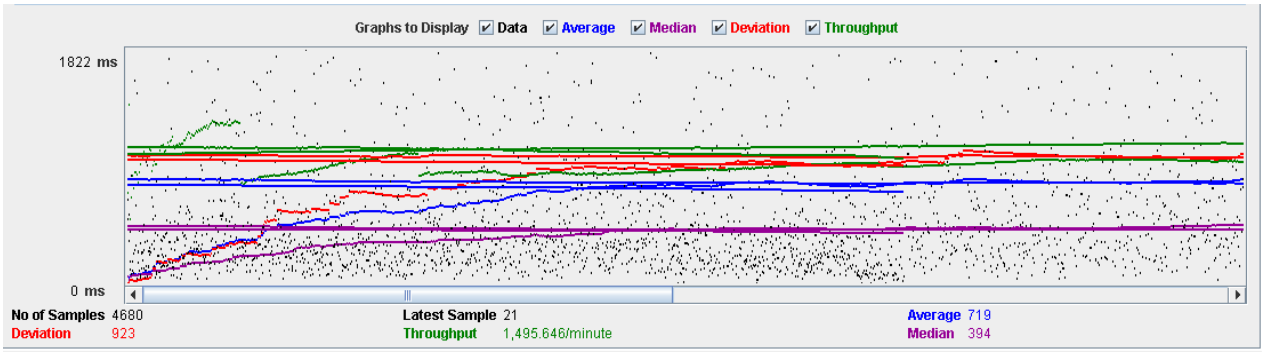
Overload test case: 26 concurrent users and 20 of loop count:

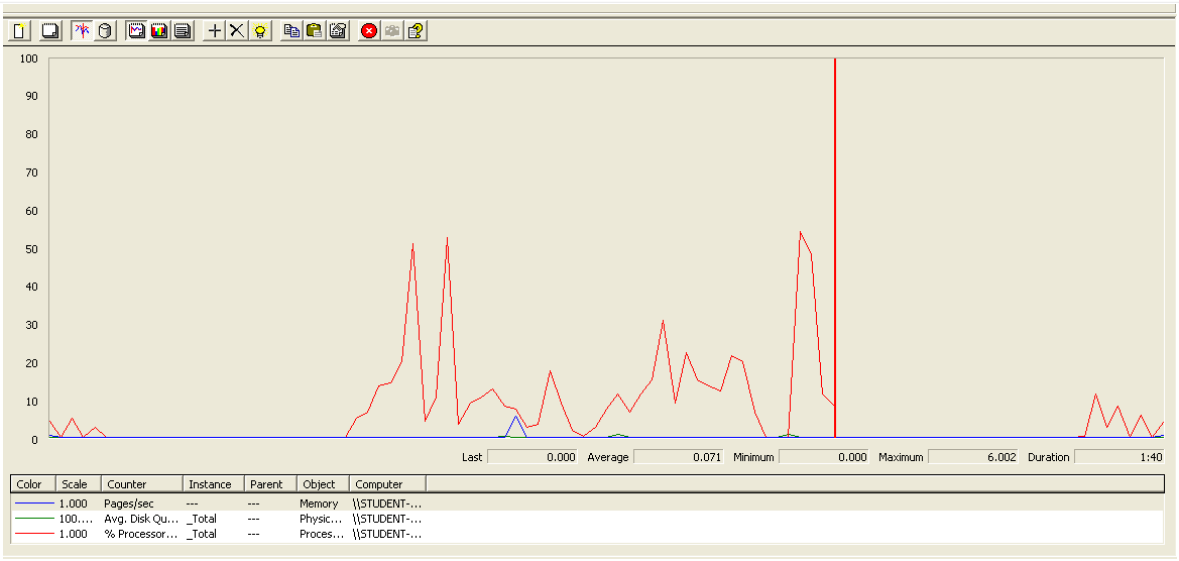


As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	520	456	196	1118	0	6986	0.00%	2.8/sec	10.6
/j_spring_security...	520	1029	599	2600	38	6663	0.00%	2.8/sec	11.0
/actor/display.do	520	552	280	1364	0	5557	0.00%	2.8/sec	15.2
/position/list.do	520	632	363	1433	30	6318	0.00%	2.8/sec	15.4
/position/display.do	520	687	405	1568	27	6872	0.00%	2.8/sec	14.0
/application/comp...	1040	680	426	1430	31	7389	0.00%	5.6/sec	26.6
/application/comp...	520	1188	645	2799	46	8924	0.00%	2.8/sec	10.8
/j_spring_security...	520	572	264	1475	5	5846	0.00%	2.8/sec	10.9
TOTAL	4680	719	394	1686	0	8924	0.00%	24.9/sec	113.4

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.





Conclusion: The maximum number of concurrent users supported by this test case is 25 and we could improve it by assigning more CPU's resources to our system.

Req. 9.3 – Manage the applications to their positions, which includes listing them grouped by status, showing them, and updating them. Updating an application amounts to making a decision on it: an application whose status is SUBMITTED may change to status ACCEPTED or REJECTED.

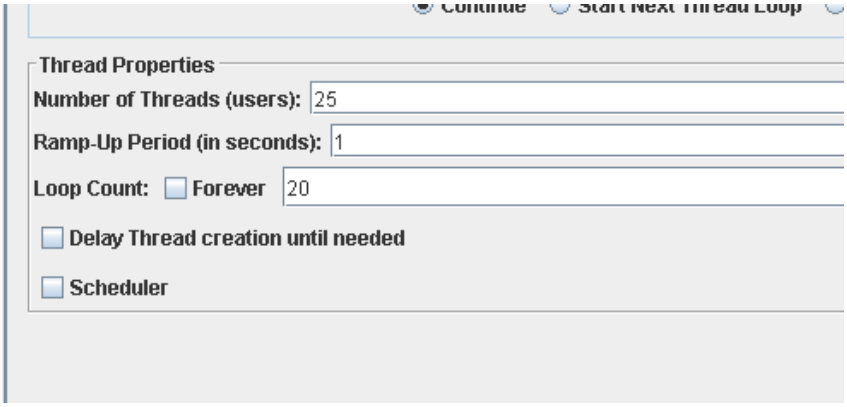
Technical details of the computer on which the test has been executed:

- Ram: 8,0 (1x) GB, DDR3 RAM (1,600 MHz)
- CPU: Intel Core i5-4200U
- Disco duro: 240 GB SSD

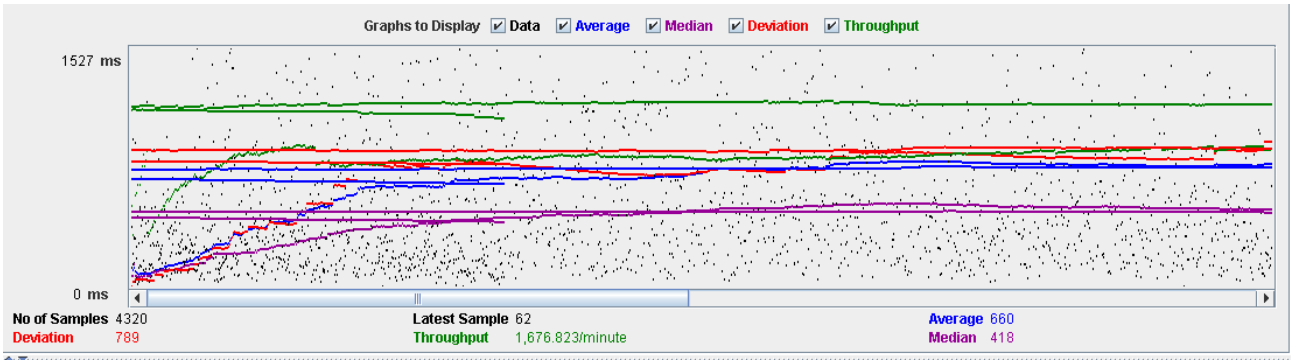
Test case description:

- The user logs in as a company.
- Display her/his profile.
- List positions.
- Display position.
- List applications.
- Accepted application.
- The company closes session.

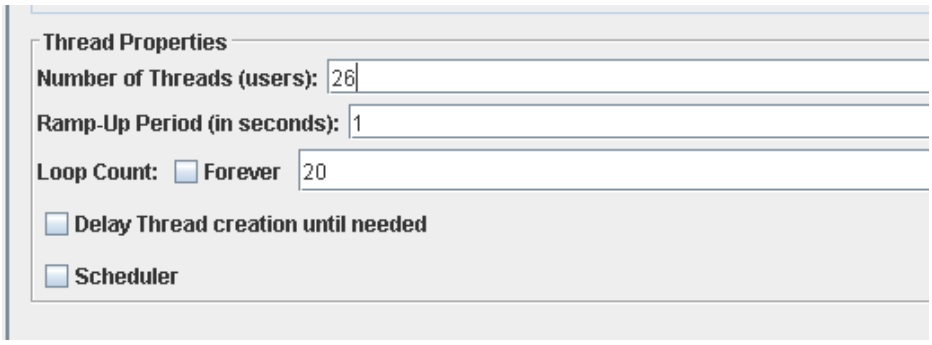
Maximum workload test case. 25 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	480	421	225	1001	7	5004	0.00%	3.1/sec	11.9
/j_spring_security_check	480	940	605	2126	43	7470	0.00%	3.1/sec	12.3
/actor/display.do	480	530	315	1172	4	5598	0.00%	3.1/sec	17.0
/position/list.do	480	541	361	1081	26	8660	0.00%	3.1/sec	17.3
/position/display.do	480	619	418	1398	15	4617	0.00%	3.2/sec	15.9
/application/company/list.do	960	668	457	1276	56	10713	0.00%	6.3/sec	30.1
/application/company/accept.do	480	1075	765	2253	64	11547	0.00%	3.2/sec	12.3
/j_spring_security_logout	480	479	265	1080	5	5746	0.00%	3.2/sec	12.1
TOTAL	4320	660	418	1435	4	11547	0.00%	27.9/sec	127.2

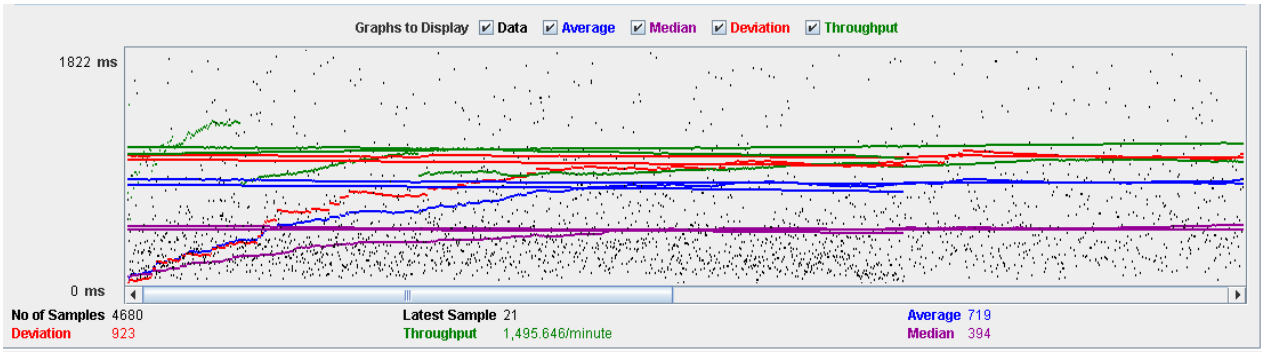


Overload test case: 26 concurrent users and 20 of loop count:

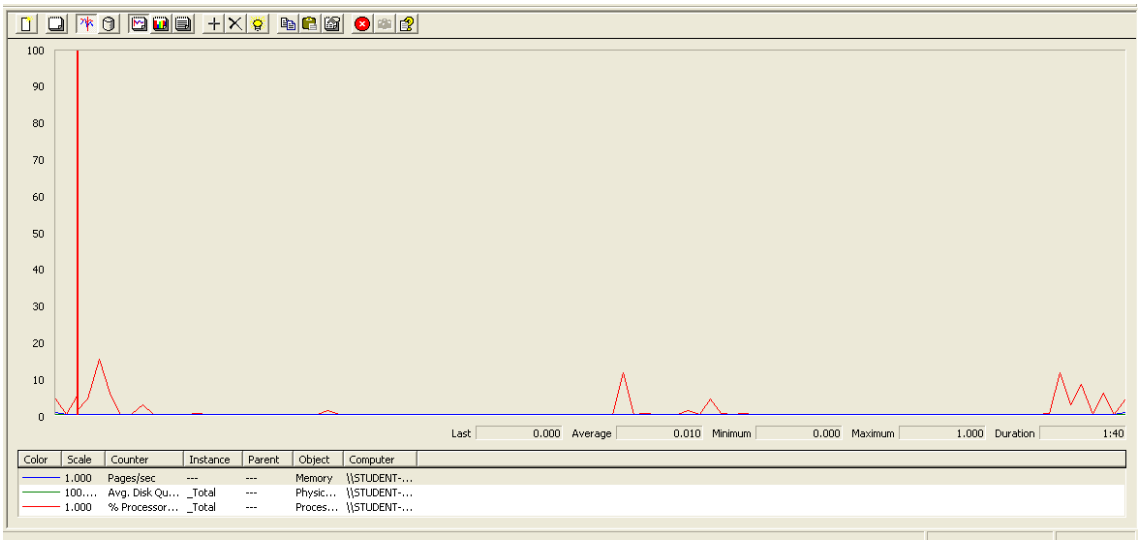


As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	520	456	196	1118	0	6986	0.00%	2.8/sec	10.6
/j_spring_security...	520	1029	599	2600	38	6663	0.00%	2.8/sec	11.0
/actor/display.do	520	552	280	1364	0	5557	0.00%	2.8/sec	15.2
/position/list.do	520	632	363	1433	30	6318	0.00%	2.8/sec	15.4
/position/display.do	520	687	405	1568	27	6872	0.00%	2.8/sec	14.0
/application/comp...	1040	680	426	1430	31	7389	0.00%	5.6/sec	26.6
/application/comp...	520	1188	645	2799	46	8924	0.00%	2.8/sec	10.8
/j_spring_security...	520	572	264	1475	5	5846	0.00%	2.8/sec	10.9
TOTAL	4680	719	394	1686	0	8924	0.00%	24.9/sec	113.4


Graphs to Display
Data
Average
Median
Deviation
Throughput
1822 ms
0 ms
No of Samples 4680
Latest Sample 21
Average 719
Deviation 923
Throughput 1,495.646/minute
Median 394

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.



Conclusion: The maximum number of concurrent users supported by this test case is 25 and we could improve it by assigning more CPU's resources to our system.

Req. 10.1 – Manage his or her applications, which includes listing them grouped by status, showing them, creating them, and updating them. When an application is created, the system assigns an arbitrary problem to it (from the set of problems that have been registered for the corresponding position). Updating an application consists in submitting a solution to the corresponding problem (a piece of text with explanations and a link to the code), registering the submission moment, and changing the status to SUBMITTED.

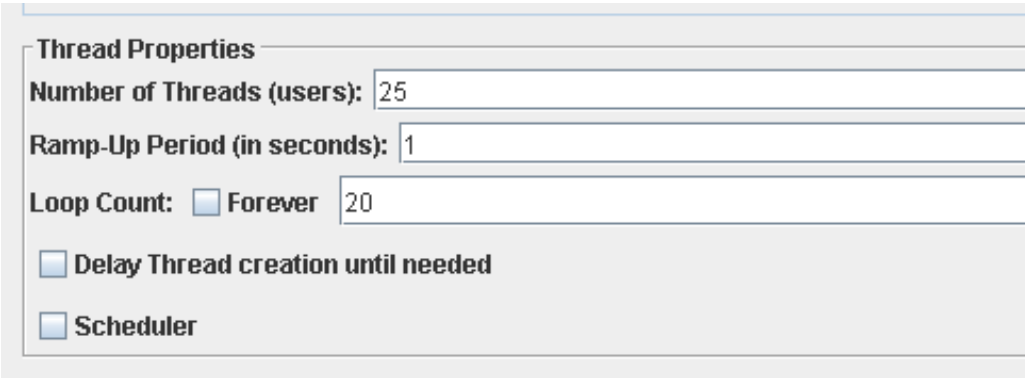
Technical details of the computer on which the test has been executed:

- Ram: 8,0 (1x) GB, DDR3 RAM (1,600 MHz)
- CPU: Intel Core i5-4200U
- Hard disk: 240 GB SSD

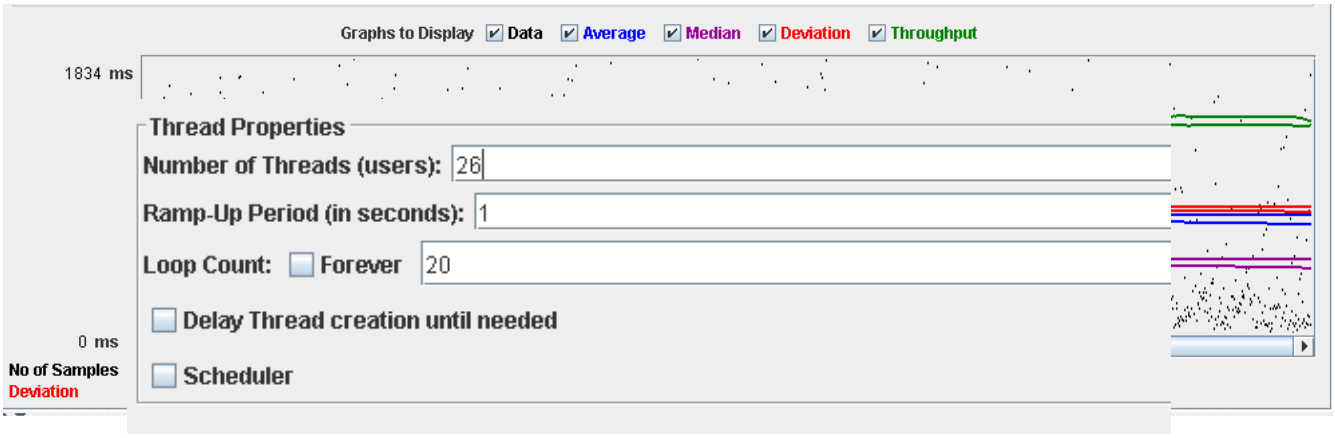
Test case description:

- The user logs in as a hacker.
- The hacker accesses the list of own applications.
- The hacker clicks on a pending application.
- The hacker displays the application's display.
- The hacker creates an answer.
- The hacker sends that answer.
- The hacker closes session.

Maximum workload test case. 25 concurrent users and 20 of loop count:



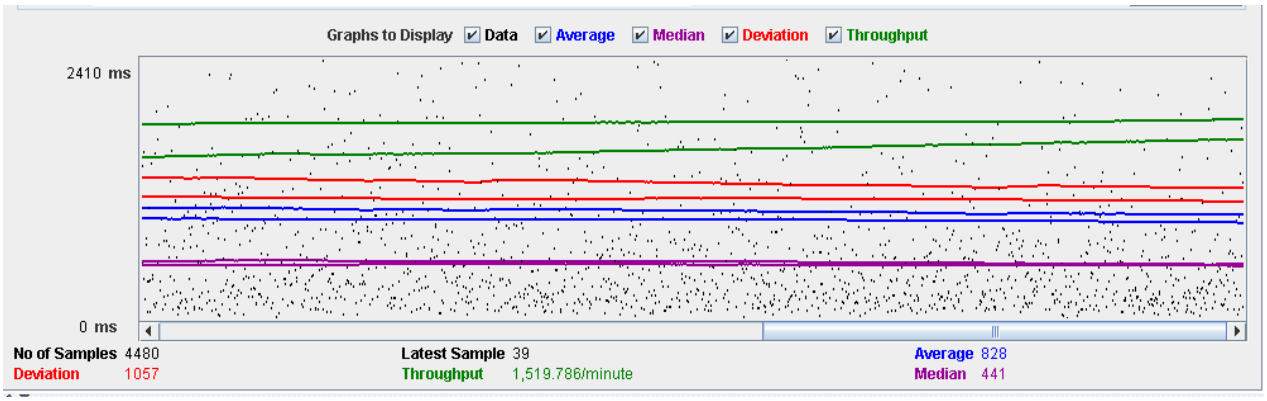
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	500	531	226	1383	0	4983	0.00%	3.4/sec	13.0
/j_spring_security_check	500	994	616	2232	66	6535	0.00%	3.4/sec	14.3
/application/hacker/list.do	1000	782	532	1650	50	6507	0.00%	6.7/sec	34.7
/application/company,hacke...	500	595	288	1506	11	6067	0.00%	3.4/sec	15.8
/answer/hacker/create.do	500	1000	643	2217	39	7223	0.00%	3.4/sec	14.2
/answer/hacker/edit.do	500	635	349	1510	18	4332	0.00%	3.4/sec	16.7
/j_spring_security_logout	500	538	242	1353	0	5247	0.00%	3.4/sec	13.1
TOTAL	4000	732	440	1743	0	7223	0.00%	26.8/sec	120.2



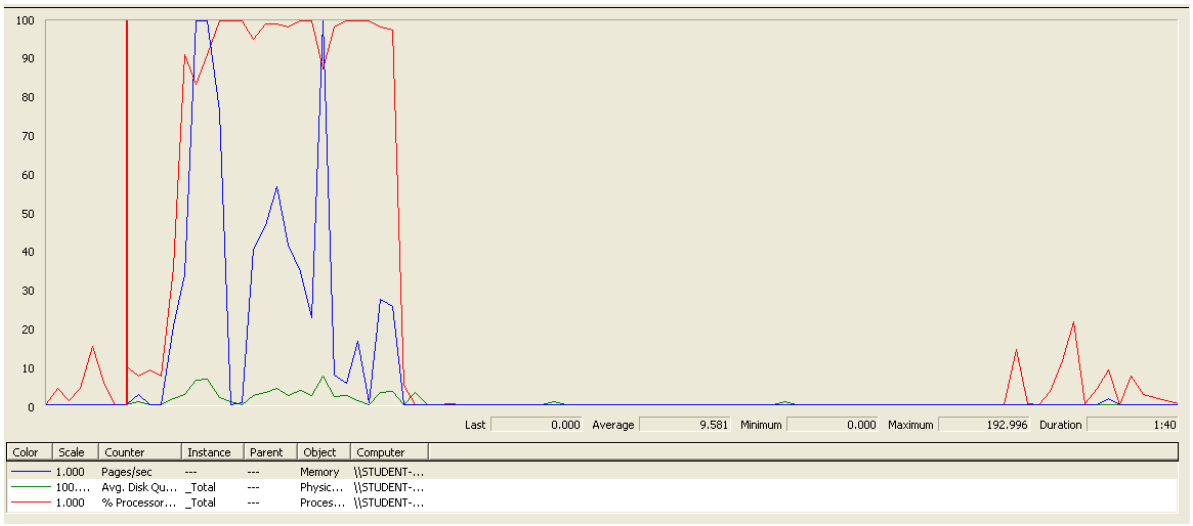
Overload test case: 26 concurrent users and 20 of loop count:

As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	560	583	236	1494	0	6514	0.00%	3.2/sec	12.2
/j_spring_security_check	560	1205	745	2893	13	11255	0.00%	3.2/sec	13.4
/application/hacker/list.do	1120	874	535	1943	65	7662	0.00%	6.4/sec	32.8
/application/company,hacke...	560	578	251	1445	0	5211	0.00%	3.2/sec	14.9
/answer/hacker/create.do	560	1217	740	2807	35	10360	0.00%	3.2/sec	13.2
/answer/hacker/edit.do	560	693	345	1722	2	11183	0.00%	3.2/sec	15.7
/j_spring_security_logout	560	599	246	1473	6	8224	0.00%	3.2/sec	12.3
TOTAL	4480	828	441	2090	0	11255	0.00%	25.3/sec	113.5



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.



Conclusion: The maximum number of concurrent users supported by this test case is 25 and we could improve it by assigning more CPU's resources to our system.

Req. 10.1 – Manage his or her applications, which includes listing them grouped by status, showing them, creating them, and updating them. When an application is created, the system assigns an arbitrary problem to it (from the set of problems that have been registered for the corresponding position). Updating an application consists in submitting a solution to the corresponding problem (a piece of text with explanations and a link to the code), registering the submission moment, and changing the status to SUBMITTED.

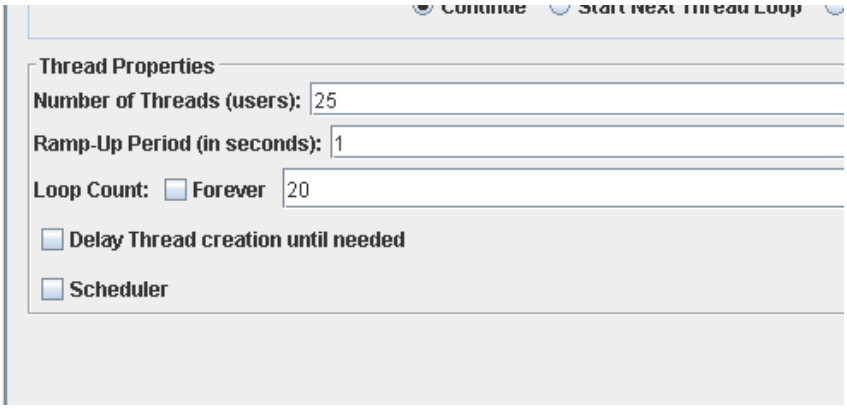
Technical details of the computer on which the test has been executed:

- Ram: 8,0 (1x) GB, DDR3 RAM (1,600 MHz)
- CPU: Intel Core i5-4200U
- Hard disk: 240 GB SSD

Test case description:

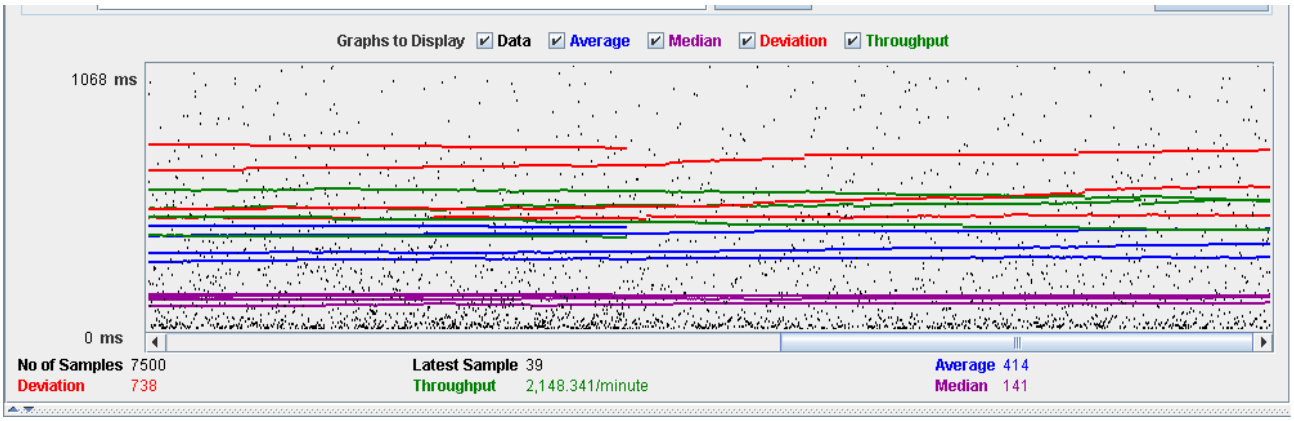
- The user logs in as a hacker.
- The hacker accesses the list of available positions.
- The hacker accesses the display of the chosen position.
- The hacker clicks to request the position.
- The hacker chooses the curriculum and saves.
- The hacker closes session.

Maximum workload test case. 25 concurrent users and 20 of loop count:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/styles/common.css	500	23	16	46	3	421	0.00%	2.4/sec	1.4
/scripts/md5-min.js	500	22	16	42	2	381	0.00%	2.4/sec	13.1
/styles/jmenu.css	500	22	14	46	4	187	0.00%	2.4/sec	4.6
/scripts/jquery.js	500	86	75	156	14	511	0.00%	2.4/sec	654.0
/styles/displaytag.css	500	19	14	40	3	159	0.00%	2.4/sec	7.3
/security/login.do	500	514	209	1291	27	4802	0.00%	2.4/sec	9.5
/j_spring_security_check	500	1049	612	2461	45	12051	0.00%	2.4/sec	10.2
/position/availableList.do	500	549	296	1195	58	6347	0.00%	2.4/sec	14.1
/position/display.do	500	746	529	1458	96	5939	0.00%	2.4/sec	11.1
/application/hacker/create.do	500	1051	669	2212	62	8505	0.00%	2.4/sec	10.1
/application/hacker/edit.do	500	632	375	1381	56	5151	0.00%	2.4/sec	29.3
/application/hacker/list.do	500	810	582	1575	120	6389	0.00%	2.4/sec	14.1
/j_spring_security_logout	500	521	255	1316	6	6663	0.00%	2.4/sec	9.3
TOTAL	7500	414	141	1086	2	12051	0.00%	35.8/sec	1883.7

Include group name in label? Save Table Data Save Table Header

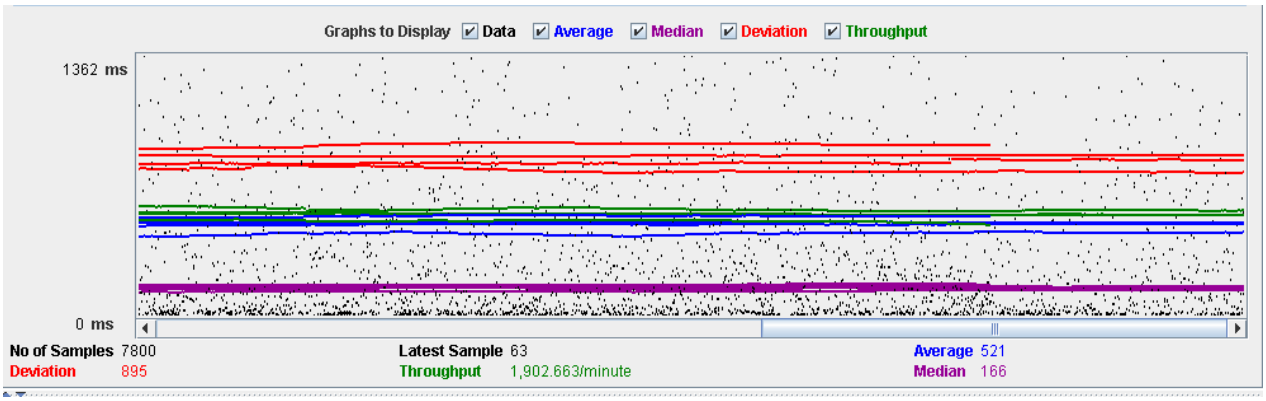


Overload test case: 26 concurrent users and 20 of loop count:

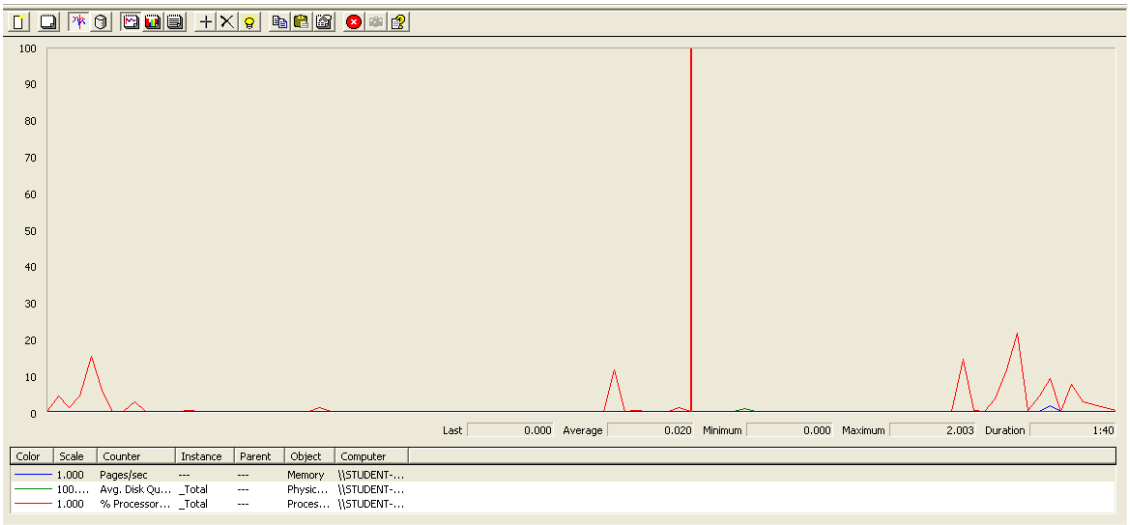
As we can see in the report if we increase the number of threads the 90% line gets a value too high.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/styles/common.css	520	25	16	60	0	858	0.00%	2.2/sec	1.2
/scripts/md5-min.js	520	21	15	59	0	305	0.00%	2.2/sec	11.6
/styles/jmenu.css	520	22	15	57	0	509	0.00%	2.2/sec	4.1
/scripts/jquery.js	520	93	84	183	0	615	0.00%	2.2/sec	580.8
/styles/displaytag.css	520	23	16	54	0	818	0.00%	2.2/sec	6.4
/security/login.do	520	662	247	1865	0	6450	0.00%	2.2/sec	8.4
/j_spring_security_check	520	1205	806	2883	23	8650	0.00%	2.2/sec	9.1
/position/availableList.do	520	744	378	1959	15	6640	0.00%	2.2/sec	12.5
/position/display.do	520	912	623	1885	102	5182	0.00%	2.2/sec	9.8
/application/hacker/create.do	520	1392	943	2972	68	11226	0.00%	2.1/sec	8.9
/application/hacker/edit.do	520	805	440	1867	58	7990	0.00%	2.2/sec	25.8
/application/hacker/list.do	520	1050	722	2197	92	10142	0.00%	2.2/sec	12.4
/j_spring_security_logout	520	683	318	1674	2	7241	0.00%	2.2/sec	8.2
TOTAL	7800	521	166	1483	0	11226	0.00%	31.7/sec	1668.3

Include group name in label? Save Table Data Save Table Header



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.



Conclusion: The maximum number of concurrent users supported by this test case is 25 and we could improve it by assigning more CPU's resources to our system.

Req. 11.1 – An actor who is authenticated as an administrator must be able to: Create user accounts for new administrators.

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

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

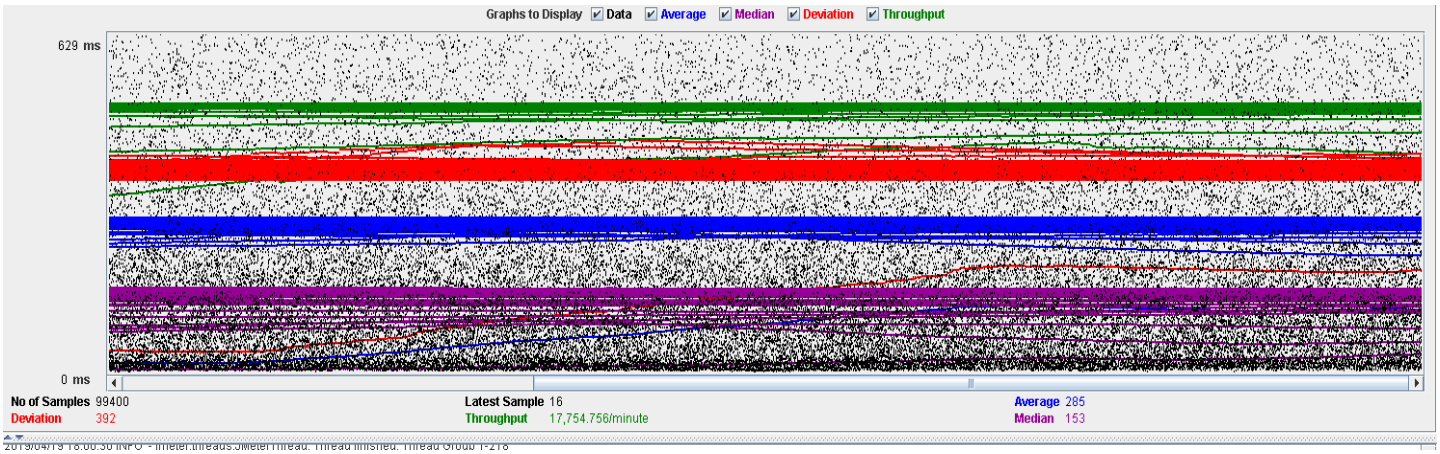
Test case description:

- The user logs in as an administrator.
- The user goes to register administrator view.
- The user completes register administrator form.
- The user logs out the system.
- The user logs in the system with the new admin account.
- The user logs out the system.

Maximum workload test case. 355 concurrent users and 20 of loop count:

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.

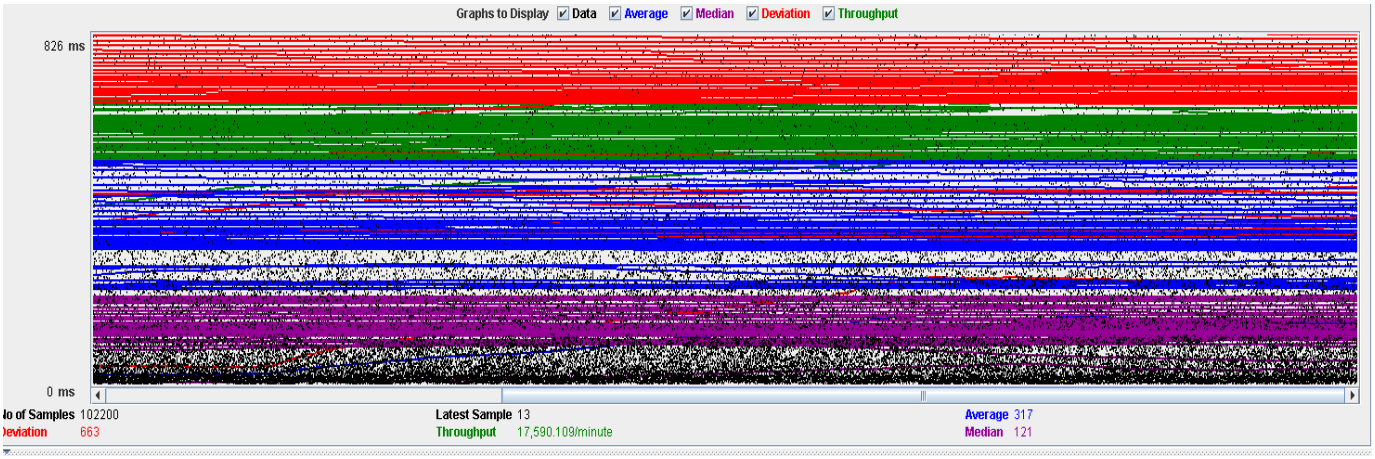
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	35500	247	129	601	3	4980	0.00%	105.7/sec	403.7
/security/login.do	14200	224	112	549	3	6416	0.00%	42.8/sec	163.8
/j_spring_security_check	14200	462	310	1035	5	7269	0.00%	42.9/sec	184.6
/actor/administrator/regi...	14200	294	160	695	4	4988	0.00%	43.6/sec	411.5
/welcome/index.do	7100	293	160	720	2	6109	0.00%	21.9/sec	89.5
/j_spring_security_logout	14200	252	135	600	4	5250	0.00%	43.3/sec	166.0
TOTAL	99400	285	153	694	2	7269	0.00%	295.9/sec	1395.3



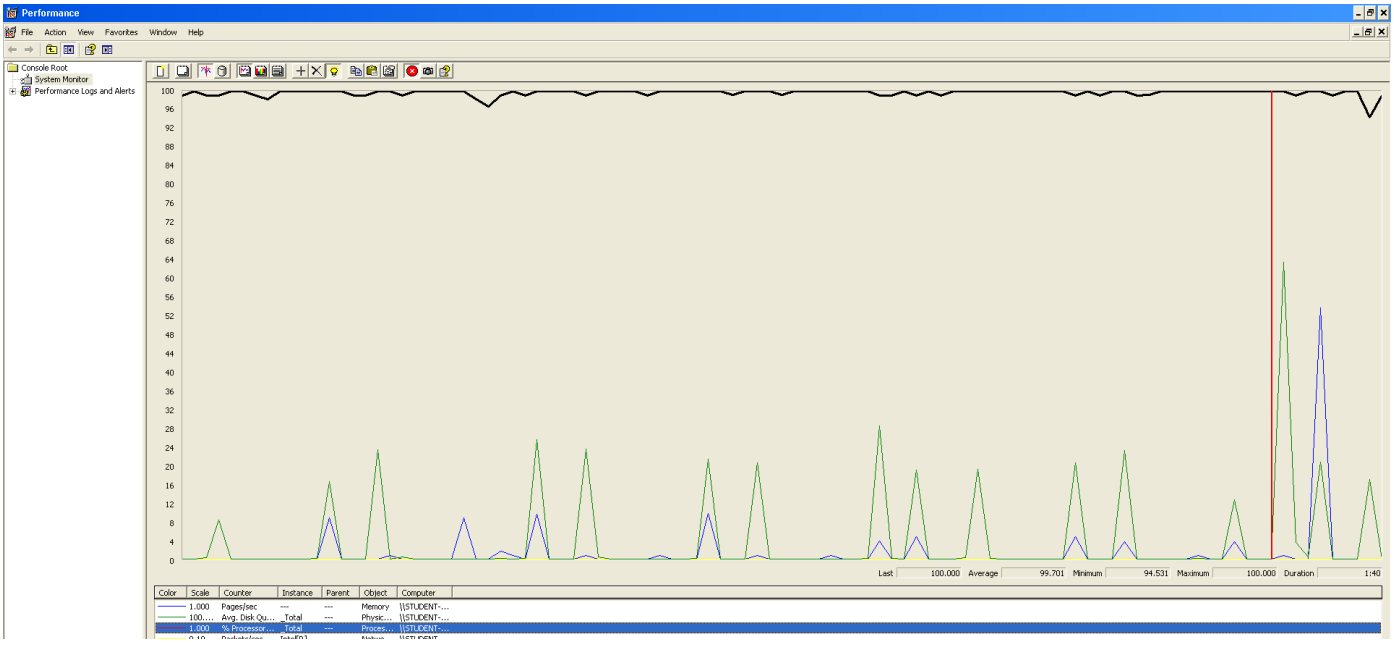
Overload test case: 365 concurrent users and 20 of loop count:

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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	36500	262	101	604	2	14652	0.01%	104.7/sec	399.9
/security/login.do	14600	255	87	596	2	22156	0.01%	42.4/sec	162.2
/j_spring_security_check	14600	533	250	1224	3	14311	0.03%	42.4/sec	182.8
/factor/administrator/regi...	14600	320	128	742	3	25612	0.00%	43.2/sec	408.0
/welcome/index.do	7300	325	124	790	3	13106	0.00%	21.7/sec	88.8
/j_spring_security_logout	14600	289	110	694	2	13011	0.01%	42.8/sec	164.3
TOTAL	102200	317	121	741	2	25612	0.01%	293.2/sec	1382.3



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.



Conclusion: The maximum number of concurrent users supported by this test case is 355 and we could improve it by assigning more CPU's resources to our system.

Req. 11.2 – An actor who is authenticated as an administrator must be able to: Display a dashboard.

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

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

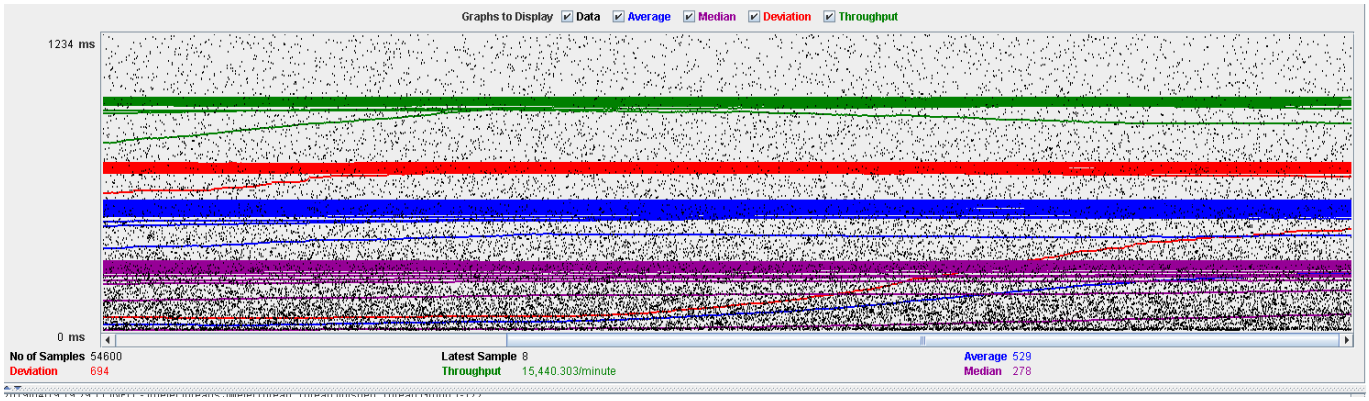
Test case description:

- The user logs in as an administrator.
- The user goes to dashboard view.
- The user logs out the system.

Maximum workload test case. 390 concurrent users and 20 of loop count:

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.

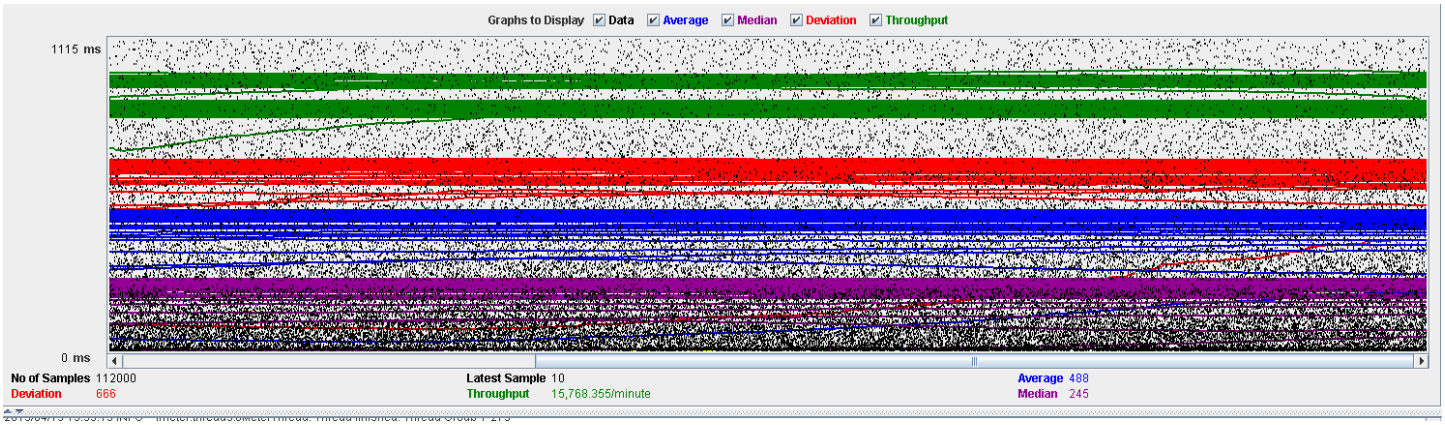
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	23400	459	233	1182	2	7223	0.00%	110.3/sec	419.3
/security/login.do	7800	443	224	1157	2	6307	0.00%	37.9/sec	145.0
/j_spring_security_check	7800	912	623	2092	6	9136	0.00%	37.8/sec	163.3
/dashboard/administrato...	7800	508	275	1249	9	8585	0.00%	37.8/sec	250.0
/j_spring_security_logout	7800	463	239	1207	5	6047	0.00%	37.8/sec	145.4
TOTAL	54600	529	278	1345	2	9136	0.00%	257.3/sec	1103.3



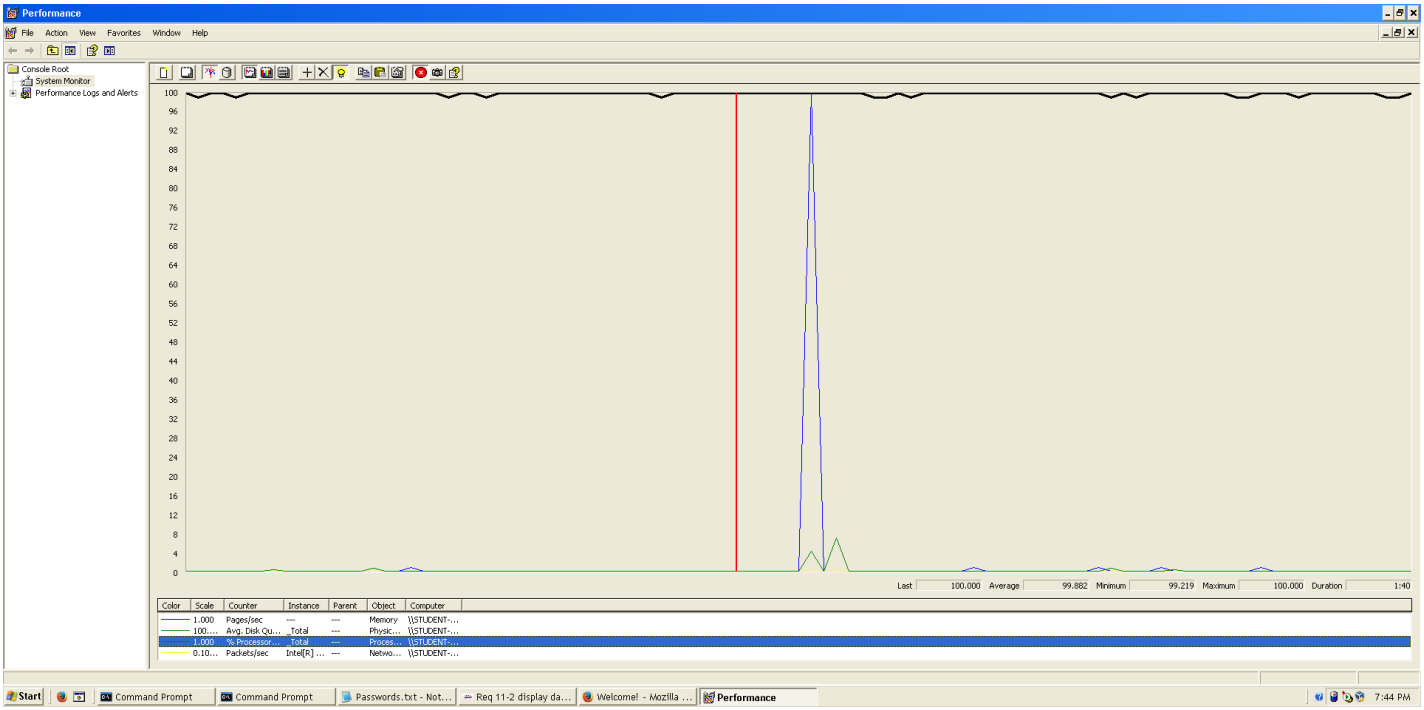
Overload test case: 400 concurrent users and 20 of loop count:

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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	48000	424	205	1103	0	8398	0.06%	112.6/sec	428.0
/security/login.do	16000	398	190	1037	2	6649	0.00%	38.1/sec	145.8
/j_spring_security_check	16000	843	552	1977	1	8513	0.11%	38.1/sec	164.3
/dashboard/administrato...	16000	463	238	1148	0	6958	0.04%	38.1/sec	251.9
/j_spring_security_logout	16000	441	220	1149	4	7879	0.00%	38.1/sec	146.3
TOTAL	112000	488	245	1261	0	8513	0.05%	262.8/sec	1126.3



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.



Conclusion: The maximum number of concurrent users supported by this test case is 390 and we could improve it by assigning more CPU's resources to our system.

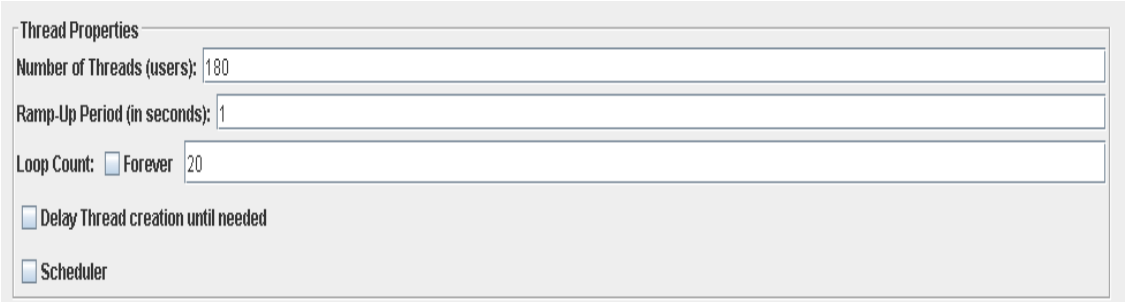
Req. 14 - The system must be easy to customise at run time. The customisation includes, but is not limited to: the name of the system (it's "Acme Hacker Rank" by default); the banner shown at the header (it's the one available at https://i.imgur.com/7b8lu4b.png by default); the message that is shown 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

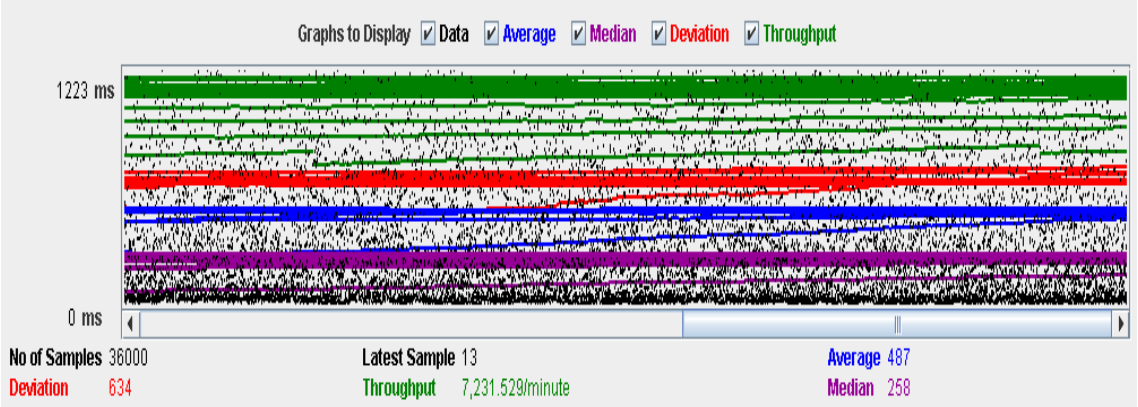
Test case description: first, an user logs in as administrator. Then, the administrator displays customisation. Later, the administrator edits customisation's parameters. Finally, the administrator logs out.

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

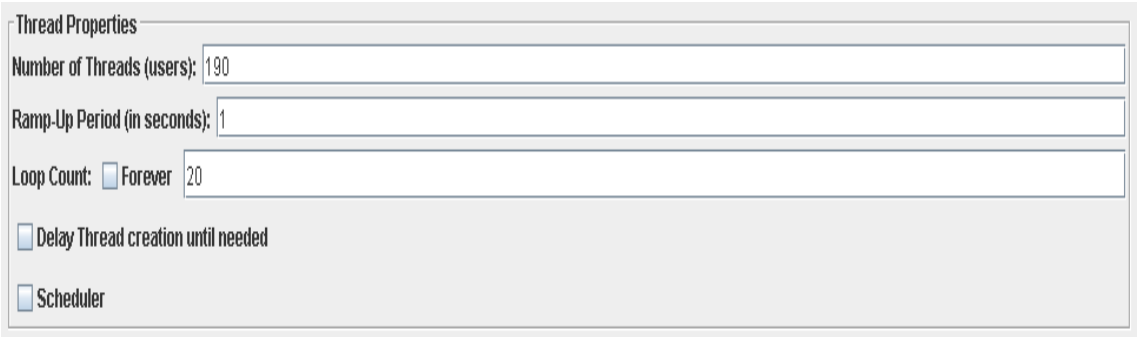


According to the performance test, this is the maximum workload that can be supported by the system without errors and failures or insufficient performance.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Through...	KB/sec
/	10800	439	229	1131	9	6534	0.00%	36.2/sec	134.5
/security/login.do	3600	419	226	1055	8	5350	0.00%	12.5/sec	47.5
/j_spring_security_check	3600	880	620	2011	19	7596	0.00%	12.5/sec	52.6
/customisation/administrator/display.do	7200	433	226	1091	12	5885	0.00%	24.7/sec	130.7
/customisation/administrator/edit.do	7200	460	227	1167	11	7435	0.00%	24.8/sec	159.7
/j_spring_security_logout	3600	471	248	1194	5	5326	0.00%	12.6/sec	47.1
TOTAL	36000	487	258	1239	5	7596	0.00%	120.5/sec	559.4

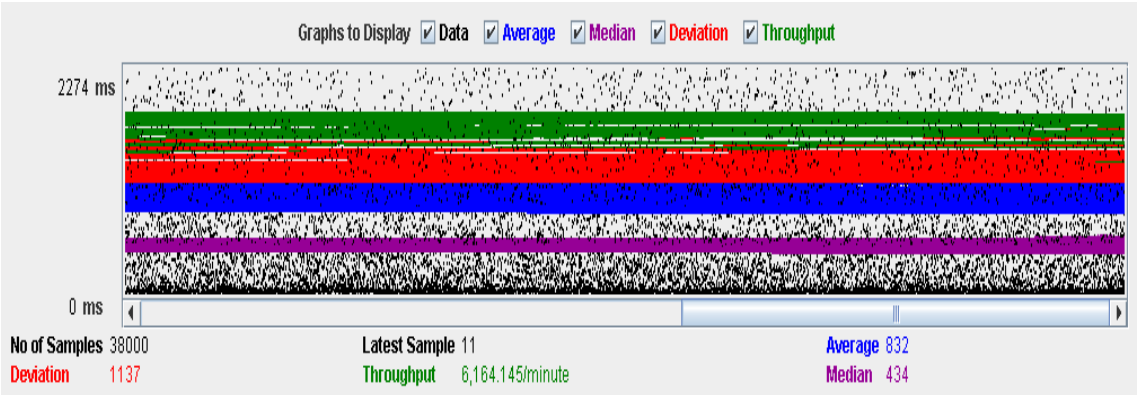


Overload test case: 190 concurrent users, 20 of loop count and 1 as ramp-up period.

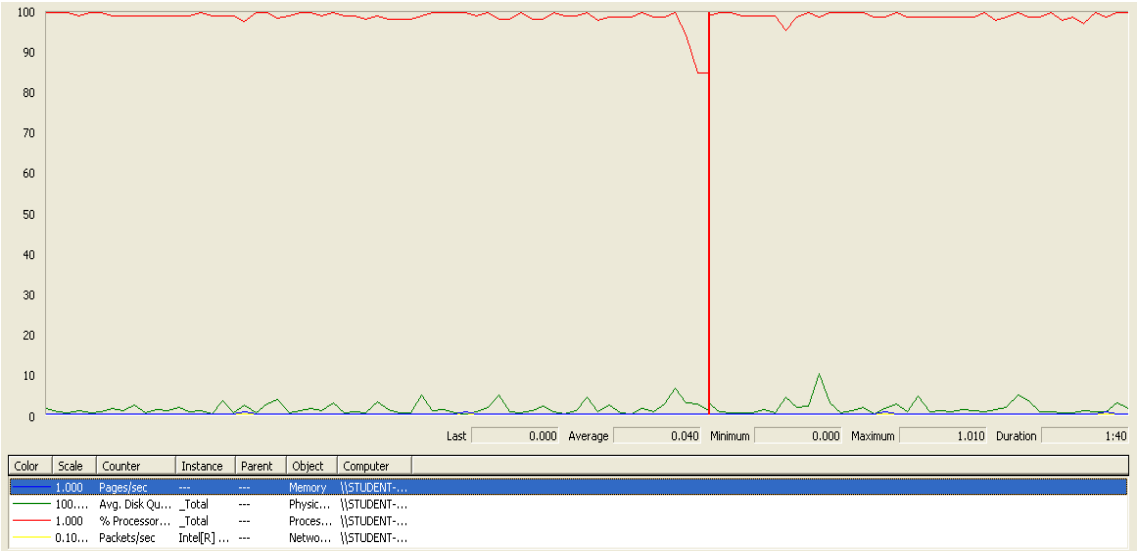


Although there aren't errors, the average time per request is not acceptable. The request with highest time is "j_spring_security_check", this is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code. Surely, that event is due to a bottleneck component.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Through...	KB/sec
/	11400	738	366	1911	8	13160	0.00%	30.8/sec	114.8
/security/login.do	3800	788	419	1952	7	9766	0.00%	10.5/sec	40.1
/j_spring_security_check	3800	1548	1087	3367	15	19481	0.00%	10.5/sec	44.5
/customisation/administrator/display.do	7600	749	382	1904	11	16416	0.00%	20.9/sec	111.0
/customisation/administrator/edit.do	7600	760	392	1937	10	14083	0.00%	21.1/sec	135.7
/j_spring_security_logout	3800	754	394	1917	6	11126	0.00%	10.6/sec	39.8
TOTAL	38000	832	434	2122	6	19481	0.00%	102.7/sec	477.3



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.



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

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
- Hard disk: 1 TB HDD - 5,400 rpm
- Interface network: Gigabit Ethernet LAN - 10BASE-T/100BASE-TX/1000BASE-T

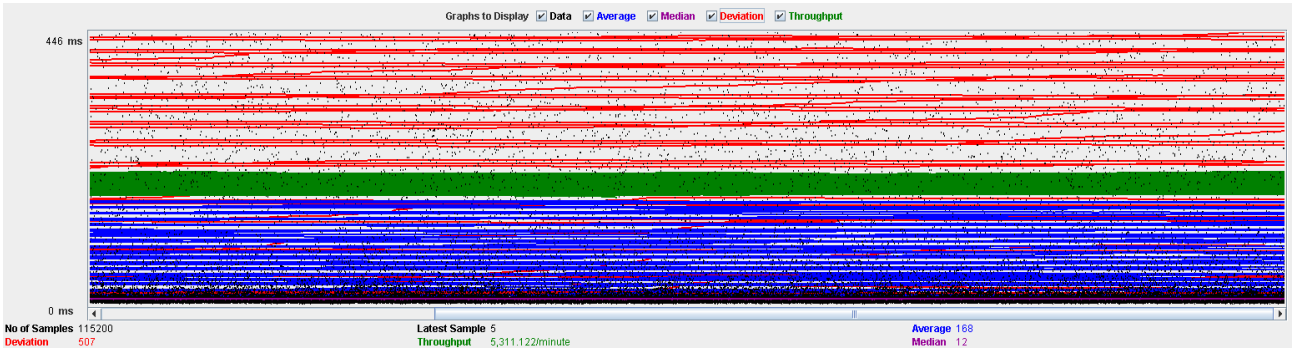
Test case description: In this test case, we test the entire use case in only one script. This is not ideal, since the tests should as specific as possible. However, since we don't use a CSV in jMeter to record the data used in the tests, the only way to test the performance of "delete" actions (delete 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

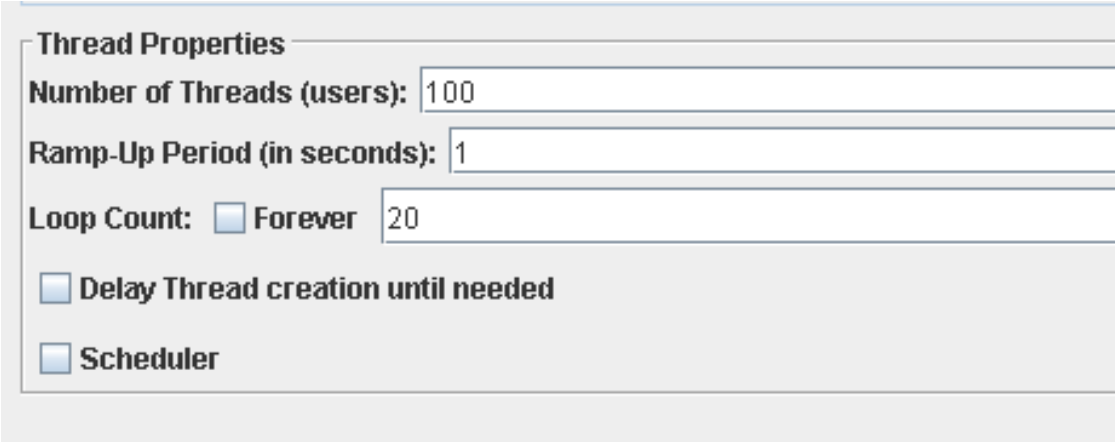
Maximum workload test case. 90 concurrent users and 20 of loop count:

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

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1800	376	47	1226	3	6944	0.00%	1.4/sec	5.3
/j_spring_security_check	1800	652	232	1922	8	5857	0.00%	1.4/sec	5.8
/	3600	301	38	923	4	7671	0.00%	2.8/sec	10.0
/curriculum/hacker/list.do	3600	963	781	1931	5	8551	0.00%	2.8/sec	17.9
/curriculum/hacker/create.do	1800	253	30	749	5	4421	0.00%	1.4/sec	7.9
/curriculum/hacker/edit.do	9000	412	48	1287	4	8754	0.00%	6.9/sec	50.7
/curriculum/display.do	27000	81	9	99	3	7978	0.00%	20.9/sec	245.0
/personalData/hacker/edit.do	3600	199	14	632	5	7773	0.00%	2.9/sec	13.5
/personalData/hacker/backCurriculum.do	1800	98	10	177	4	4453	0.00%	1.4/sec	17.0
/positionData/hacker/create.do	3600	64	9	48	3	4322	0.00%	2.9/sec	14.1
/positionData/hacker/edit.do	10800	64	12	47	5	4609	0.00%	8.6/sec	40.7
/positionData/hacker/backCurriculum.do	5400	41	9	28	4	7081	0.00%	4.3/sec	50.5
/educationData/hacker/create.do	3600	42	8	17	3	4819	0.00%	2.9/sec	14.5
/educationData/hacker/edit.do	10800	50	11	24	5	7298	0.00%	8.6/sec	41.5
/educationData/hacker/backCurriculum.do	5400	34	8	20	4	5133	0.00%	4.3/sec	50.6
/miscellaneousData/hacker/create.do	3600	130	9	319	4	7184	0.00%	2.9/sec	13.4
/miscellaneousData/hacker/edit.do	10800	177	13	478	4	11095	0.00%	8.6/sec	39.3
/miscellaneousData/hacker/backCurriculu...	5400	129	10	291	3	6056	0.00%	4.3/sec	50.6
/j_spring_security_logout	1800	353	65	989	6	7622	0.00%	1.4/sec	5.2
TOTAL	115200	168	12	490	3	11095	0.00%	88.5/sec	678.9



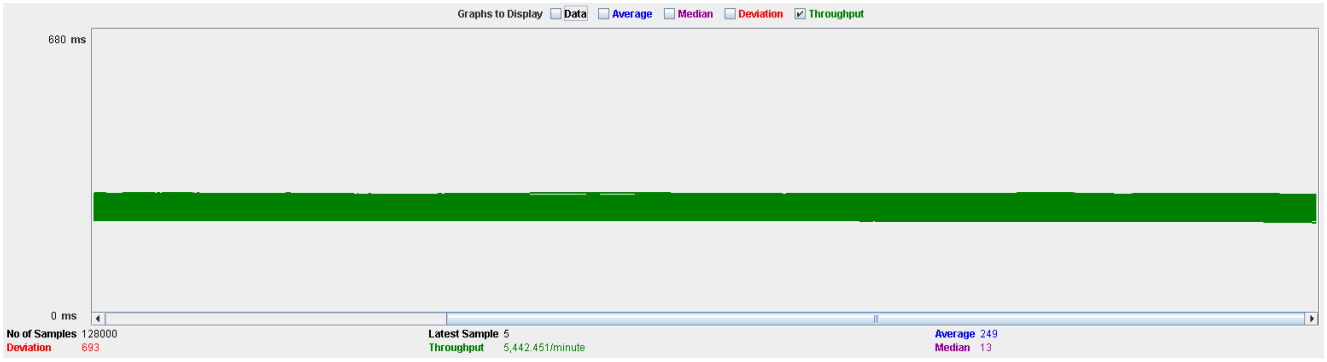
Overload test case: 100 concurrent users and 20 of loop count:



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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	2000	482	111	1454	3	7649	0.00%	1.5/sec	5.5
/j_spring_security_check	2000	924	383	2618	7	10647	0.00%	1.5/sec	6.0
/	4000	415	81	1224	4	10166	0.00%	2.8/sec	10.3
/curriculum/hacker/list.do	4000	1175	946	2374	5	8973	0.00%	2.8/sec	18.3
/curriculum/hacker/create.do	2000	390	63	1190	5	7438	0.00%	1.5/sec	8.1
/curriculum/hacker/edit.do	10000	618	109	1902	2	25229	0.00%	7.1/sec	51.9
/curriculum/display.do	30000	133	11	253	3	11847	0.00%	21.4/sec	251.0
/personalData/hacker/edit.do	4000	336	18	1078	5	10020	0.00%	2.9/sec	13.8
/personalData/hacker/backCurriculum.do	2000	175	13	453	4	4405	0.00%	1.5/sec	17.4
/positionData/hacker/create.do	4000	112	10	198	4	9217	0.00%	2.9/sec	14.4
/positionData/hacker/edit.do	12000	111	13	169	4	8150	0.00%	8.8/sec	41.6
/positionData/hacker/backCurriculum.do	6000	72	10	77	3	8729	0.00%	4.4/sec	51.6
/educationData/hacker/create.do	4000	63	9	30	3	6537	0.00%	2.9/sec	14.8
/educationData/hacker/edit.do	12000	76	12	38	4	9202	0.00%	8.8/sec	42.4
/educationData/hacker/backCurriculum.do	6000	52	9	28	4	5618	0.00%	4.4/sec	51.6
/miscellaneousData/hacker/create.do	4000	220	10	609	4	8257	0.00%	2.9/sec	13.7
/miscellaneousData/hacker/edit.do	12000	282	15	872	5	11606	0.00%	8.8/sec	40.2
/miscellaneousData/hacker/backCurriculu...	6000	191	12	535	4	8019	0.00%	4.4/sec	51.7
/j_spring_security_logout	2000	484	107	1356	6	10824	0.00%	1.5/sec	5.4
TOTAL	128000	249	13	770	2	25229	0.00%	90.7/sec	695.7



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%.

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.

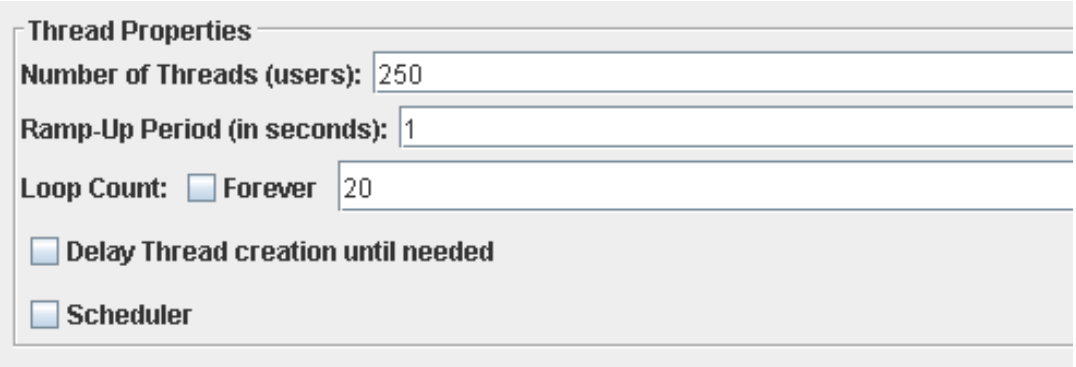
Technical details of the computer on which the test has been executed:

- Ram: 8,192 (1x) MB, DDR3L RAM (1,600 MHz)
- CPU: Intel Core i7-5500U
- Hard disk: 1 TB HDD - 5,400 rpm
- Interface network: Gigabit Ethernet LAN - 10BASE-T/100BASE-TX/1000BASE-T

Test case description:

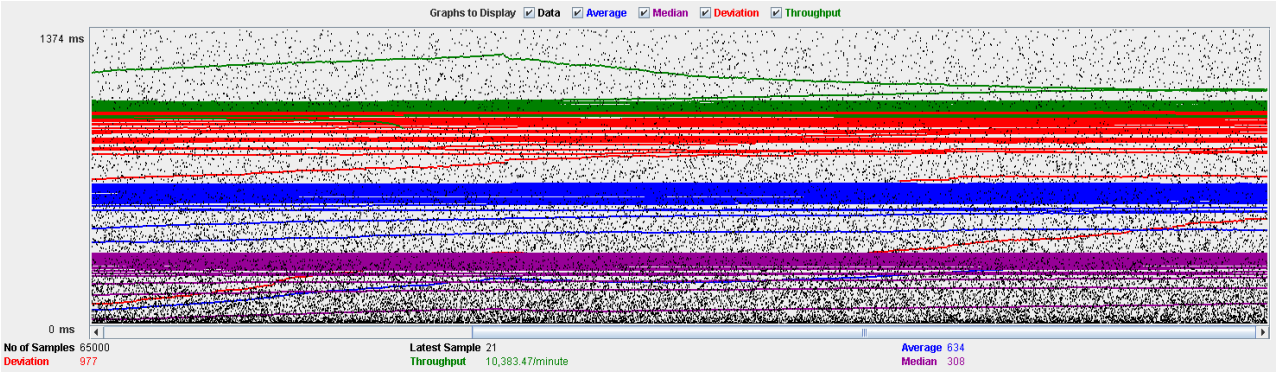
- Log in as hacker
- Go to Finder view
- Clear the Finder
- Edit Finder with key word “56”, maximum deadline “12/12/2019” and minimum salary “5.0”.
- Log out

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

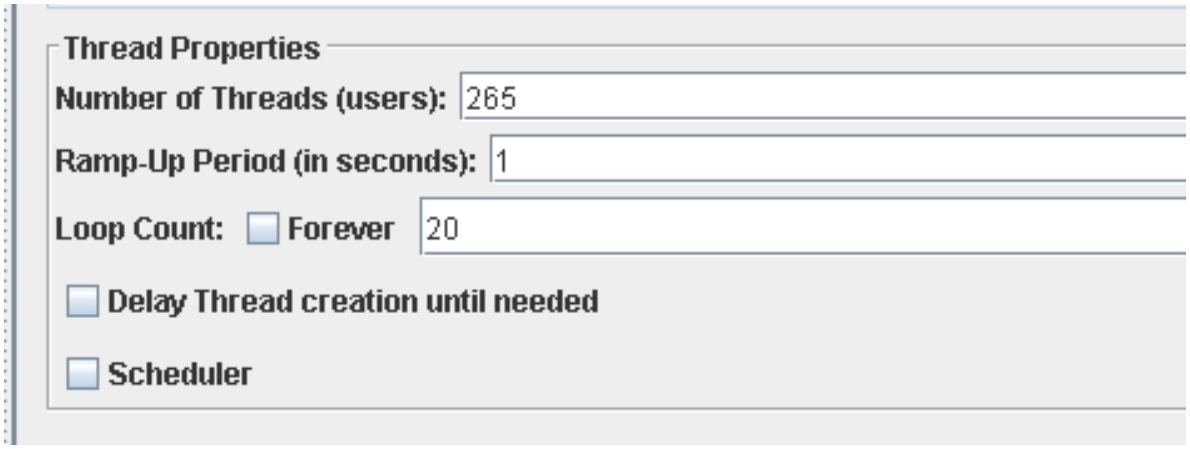


This is the maximum workload of the test case without any crash or excessive delay. As we can see in the picture below, we don’t have any errors and the average time per request is high for some requests, but still acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	15000	542	270	1331	3	11177	0.00%	39.9/sec	150.6
/security/login.do	5000	564	278	1344	3	15057	0.00%	13.6/sec	52.2
/j_spring_security_check	5000	1129	751	2401	8	18709	0.00%	13.6/sec	57.7
/finder/hacker/display.do	15000	591	298	1436	7	13429	0.00%	40.4/sec	305.8
/finder/hacker/clear.do	5000	1018	666	2314	11	14530	0.00%	13.7/sec	137.5
/images/arrow_off.png	5000	4	4	7	0	110	0.00%	13.7/sec	6.1
/finder/hacker/edit.do	10000	762	416	1793	6	14482	0.00%	27.2/sec	161.8
/j_spring_security_logout	5000	602	292	1481	5	14381	0.00%	13.7/sec	52.5
TOTAL	65000	634	308	1586	0	18709	0.00%	173.1/sec	909.4



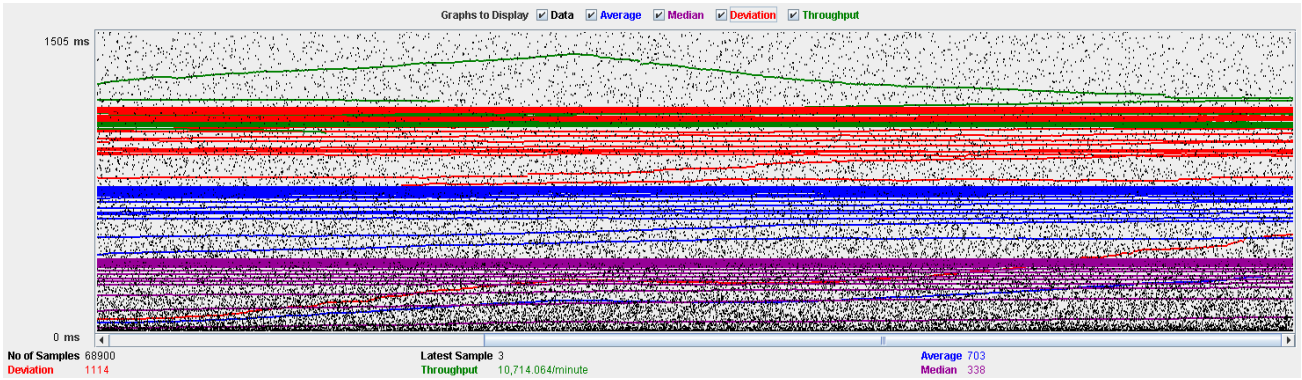
Overload test case: 265 concurrent users and 20 of loop count:



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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	15900	598	294	1468	3	14758	0.00%	41.2/sec	155.5
/security/login.do	5300	641	299	1512	4	26089	0.00%	14.1/sec	53.8
/j_spring_security_check	5300	1234	807	2667	8	22090	0.00%	14.1/sec	59.5
/finder/hacker/display.do	15900	649	328	1552	7	26118	0.00%	41.7/sec	314.8
/finder/hacker/clear.do	5300	1154	750	2588	10	29130	0.00%	14.1/sec	138.9
/images/arrow_off.png	5300	5	4	7	0	240	0.00%	14.1/sec	6.3
/finder/hacker/edit.do	10600	871	475	2123	5	19319	0.00%	28.0/sec	167.4
/j_spring_security_logout	5300	620	314	1526	6	14824	0.00%	14.1/sec	54.2
TOTAL	68900	703	338	1744	0	29130	0.00%	178.6/sec	935.8



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.

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.

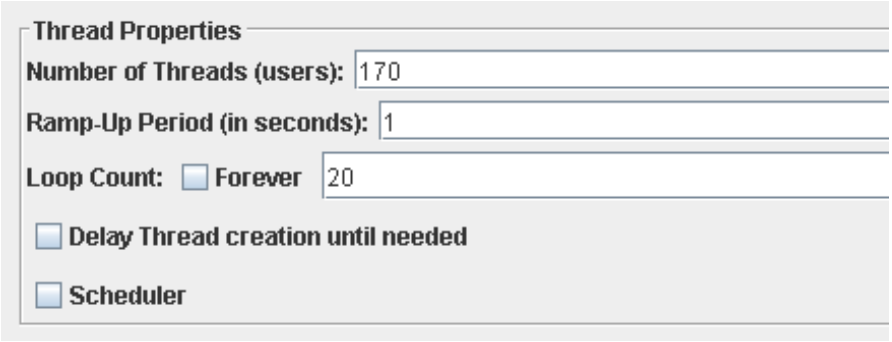
Req. 21 - The actors of the system can register their social profiles. The system must store the following data regarding them: a nick, the name of the social network, a link to a profile in that social network.
Req. 23.1 - An actor who is authenticated must be able to: Manage his or her social profiles, 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

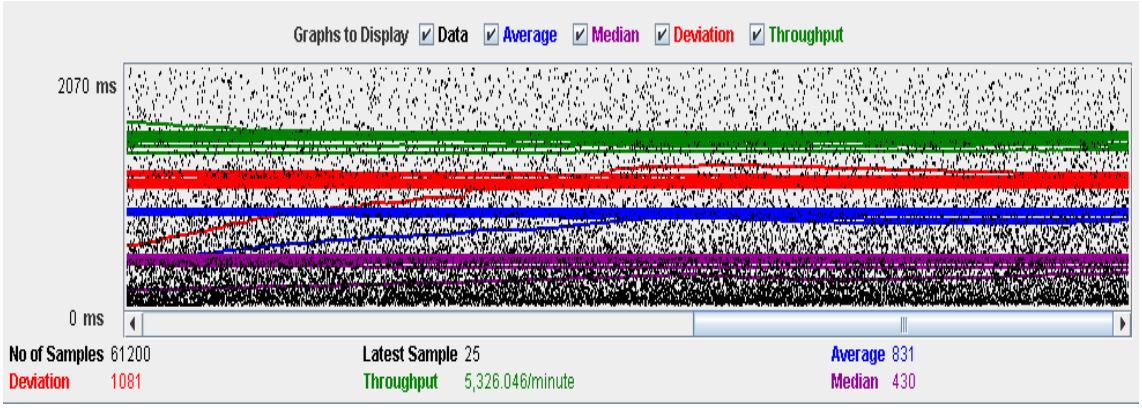
Test case description: first, an authenticated user logs in to the system. Next, he or she goes to his or her profile. Later, the user lists their social profiles and creates a new social profile. Below, the user edits the newly social profile. Then, a social profile is displayed. Immediately, the edited 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:

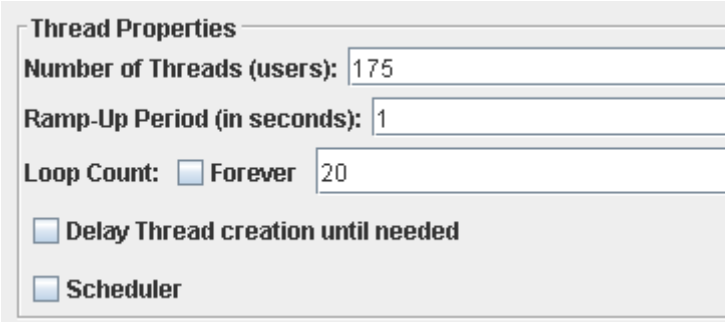


As we can check with the next report, there are not errors and the performance is acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throu...	KB/sec
/security/login.do	3400	459	127	1138	7	13339	0.00%	4.9/sec	19.1
/j_spring_security_check	3400	931	376	2441	12	17462	0.00%	4.9/sec	20.6
/	6800	468	142	1151	9	14743	0.00%	9.5/sec	36.0
/actor/display.do	3400	472	166	1070	13	15528	0.00%	4.9/sec	26.8
/socialProfile/list.do	17000	931	366	2478	15	23324	0.00%	23.9/s...	99.0
/socialProfile/display.do	3400	893	363	2445	19	14276	0.00%	4.9/sec	20.3
/socialProfile/administrator,company,hacker/create.do	3400	524	169	1218	17	17129	0.00%	4.9/sec	24.5
/socialProfile/administrator,company,hacker/edit.do	17000	746	278	1890	11	17819	0.00%	24.1/s...	142.3
/j_spring_security_logout	3400	478	153	1170	13	12563	0.00%	4.9/sec	18.7
TOTAL	61200	727	258	1876	7	23324	0.00%	85.1/s...	399.2

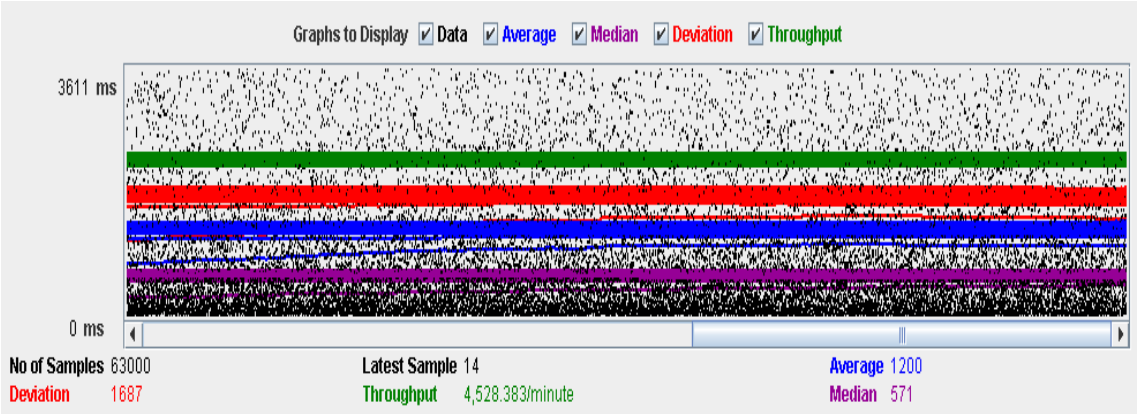


Overload test case: 175 concurrent users and 20 of loop count and 1 as ramp-up period:

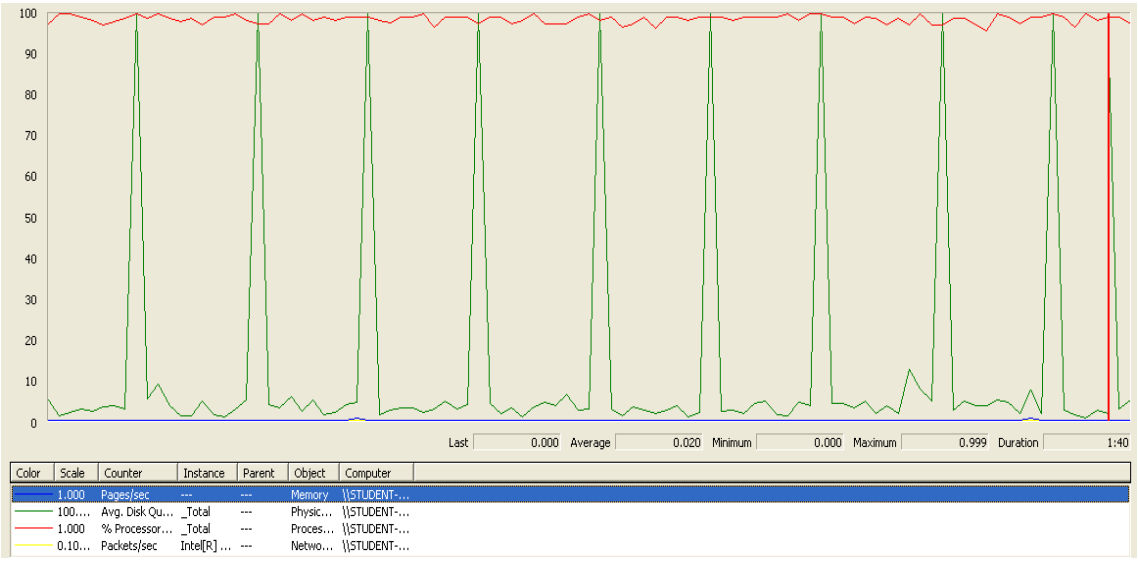


Even though it doesn't produce any errors, the average time per request is not acceptable. It very possible that a component or several components are bottleneck.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	3500	732	265	1924	8	19333	0.00%	4.3/sec	16.7
/j_spring_security_check	3500	1509	876	3734	19	16308	0.00%	4.3/sec	18.4
/	7000	784	297	2084	10	22254	0.00%	8.4/sec	32.3
/actor/display.do	3500	807	325	2044	17	16568	0.00%	4.3/sec	23.7
/socialProfile/list.do	17500	1559	874	3864	16	29169	0.00%	21.1/sec	89.4
/socialProfile/administrator,c...	3500	775	329	2034	16	10854	0.00%	4.3/sec	21.8
/socialProfile/administrator,c...	17500	1234	607	3194	10	20337	0.00%	21.2/sec	127.1
/socialProfile/display.do	3500	1499	837	3704	18	21700	0.00%	4.3/sec	18.1
/j_spring_security_logout	3500	753	290	2058	10	13246	0.00%	4.3/sec	16.5
TOTAL	63000	1200	571	3150	8	29169	0.00%	75.5/sec	359.3



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%.



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.

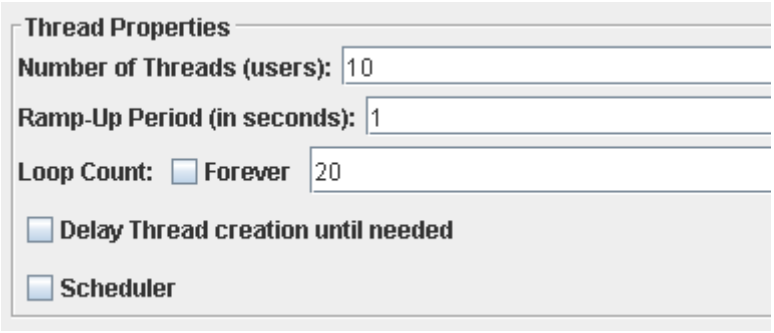
Req. 22 - Every actor has a pool of messages. For every message, the system must keep track of the sender, the recipient, the moment when it was sent, the subject, its body, and some optional tags.
Req. 23.2 - Manage his or her messages, which includes listing them grouped by tag, showing them, sending 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

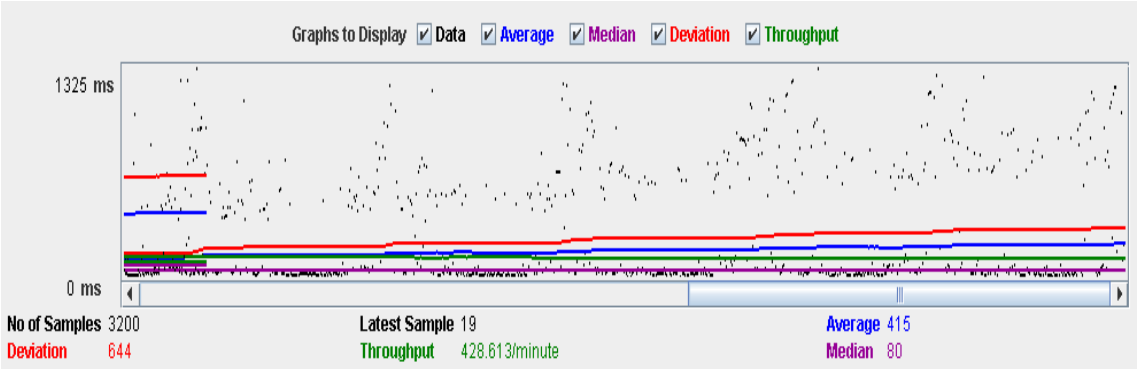
Test case description: first, an authenticated user logs in to the system. Then, the user lists their messages. Immediately, he or she sends a message. Next, a message is displayed. Later, a message is tagged as "DELETED" by the user. After, the message is deleted. Lastly, the user logs out. As happened 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:

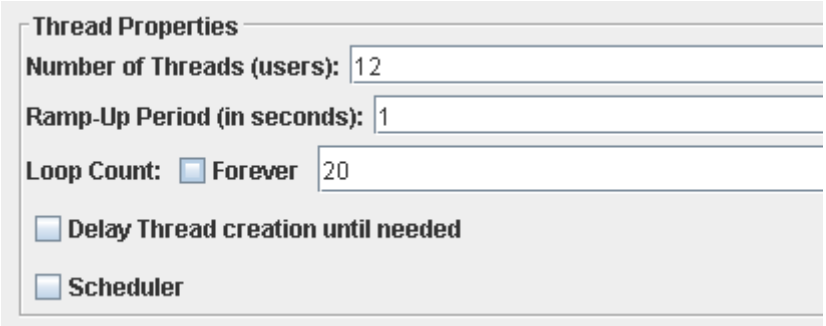


The below reports allow us to check that this is the maximum performance of the system. There are not errors and the performance is good.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throu...	KB/sec
/	600	43	25	64	7	1500	0.00%	1.3/sec	5.0
/security/login.do	200	31	18	40	9	665	0.00%	28.9/...	1.8
/j_spring_security_check	200	93	43	130	15	1913	0.00%	28.9/...	2.0
/message/administrator,company,hacker/list.do	1000	922	720	2010	57	4058	0.00%	2.3/sec	18.8
/message/administrator,company,hacker/send.do	400	637	256	1858	34	4251	0.00%	56.4/...	6.8
/message/administrator,company,hacker/display.do	200	152	45	290	22	2066	0.00%	28.0/...	1.9
/message/administrator,company,hacker/delete.do	400	146	48	354	19	2860	0.00%	55.3/...	3.8
/j_spring_security_logout	200	58	30	95	11	708	0.00%	27.7/...	1.8
TOTAL	3200	415	80	1402	7	4251	0.00%	7.1/sec	40.9

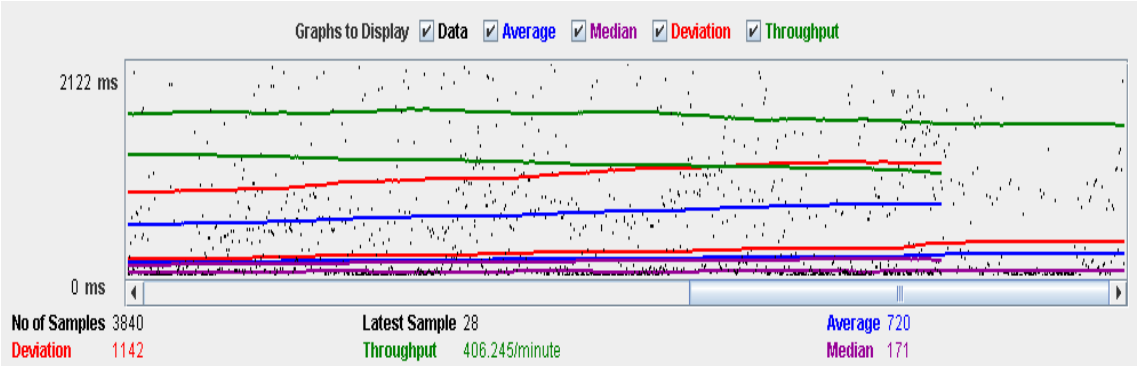


Overload test case: 12 concurrent users, 20 of loop count and 1 as ramp-up period:

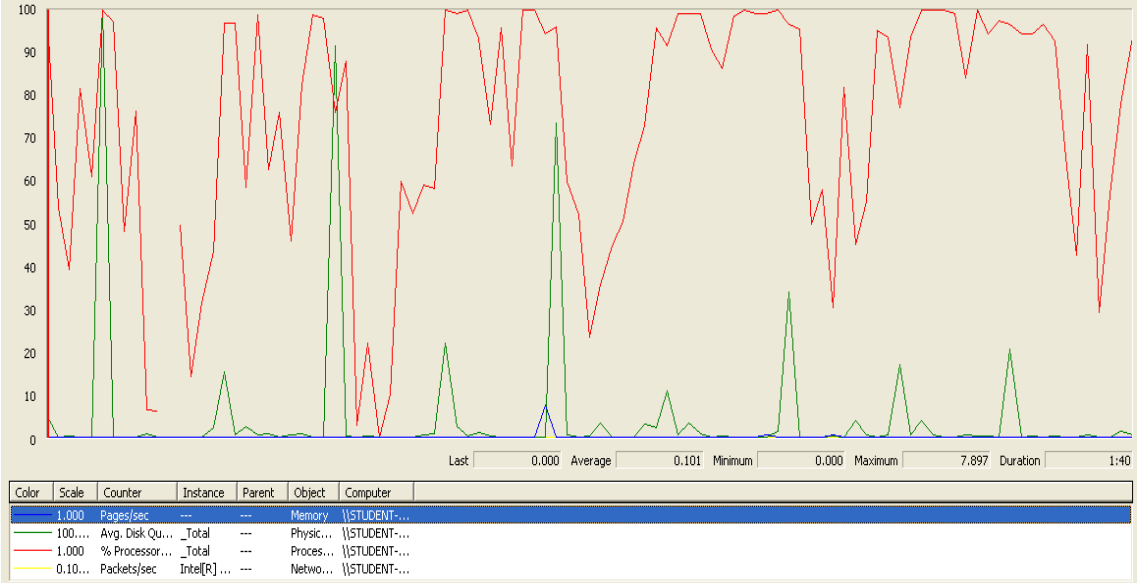


We have not any errors, but the system works with excessive delay. The system takes so much when the messages are listed and a message is sent. Analysing the code, the method MessageService::messageIsSpam is very inefficient due to we must check word by word that the message doesn't contain a spam word.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

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throu...	KB/sec
/	720	125	26	185	6	4225	0.00%	1.3/sec	4.8
/security/login.do	240	164	19	591	7	2814	0.00%	26.6/...	1.7
/j_spring_security_check	240	278	46	802	12	5636	0.00%	26.6/...	1.9
/message/administrator,company,hacker/list.do	1200	1431	1070	2965	58	10092	0.00%	2.1/sec	17.8
/message/administrator,company,hacker/send.do	480	1026	418	2849	21	8841	0.00%	52.6/...	6.4
/message/administrator,company,hacker/display.do	240	441	105	1329	20	4648	0.00%	26.3/...	1.8
/message/administrator,company,hacker/delete.do	480	405	81	1260	13	6306	0.00%	52.3/...	3.6
/j_spring_security_logout	240	238	36	644	14	4556	0.00%	26.1/...	1.7
TOTAL	3840	720	171	2186	6	10092	0.00%	6.8/sec	38.8



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.



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).

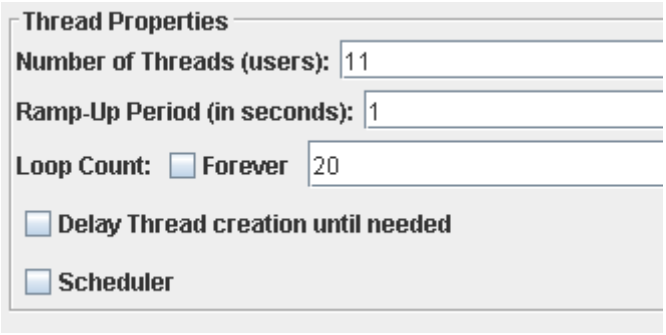
Req. 24.1 - Broadcast a notification message to the actors of the system. The message must have tag “SYSTEM” by default.

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

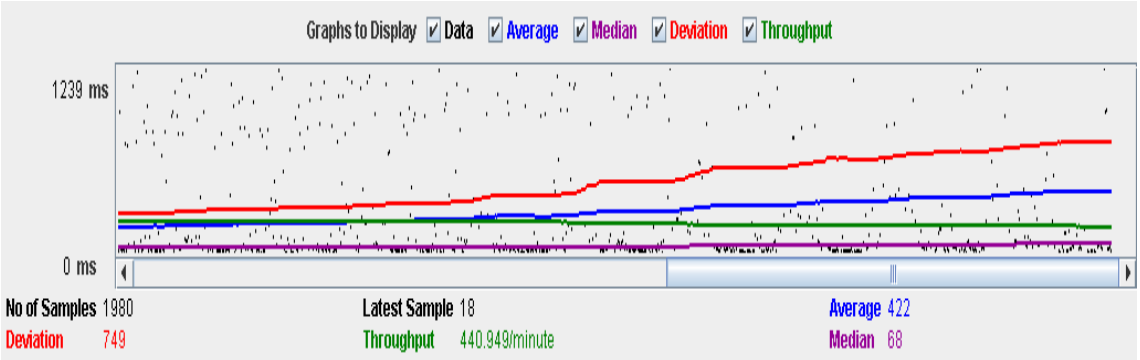
Test case description: first, an administrator logs in to the system. Then, an administrator goes to his or her messages list. Later, an administrator broadcasts a notification message. Finally, the administrator logs out.

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

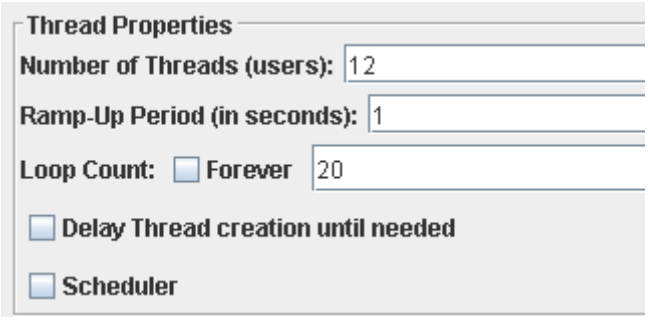


This is the maximum performance of the system without errors and failures or notable delays.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	220	47	18	61	8	1519	0.00%	51.2/min	3.3
/j_spring_security_check	220	104	43	131	19	2718	0.00%	51.2/min	3.7
/	440	50	24	72	7	1610	0.00%	1.6/sec	6.3
/message/administrator,company,hacker/list.do	440	934	698	2089	17	4895	0.00%	1.7/sec	12.0
/message/administrator/broadcast.do	440	784	392	2240	27	6185	0.00%	1.7/sec	10.8
/j_spring_security_logout	220	111	31	192	13	3049	0.00%	50.4/min	3.2
TOTAL	1980	422	68	1299	7	6185	0.00%	7.3/sec	38.4

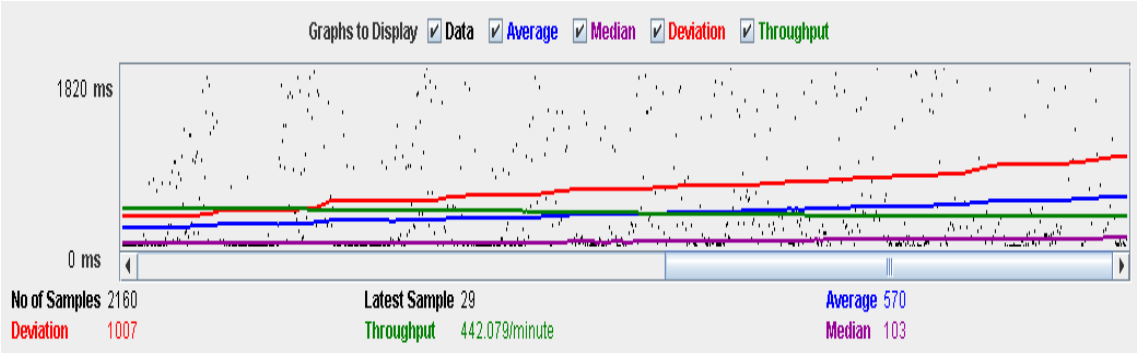


Overload test case: 12 concurrent users, 20 of loop count and 1 as ramp-up period:

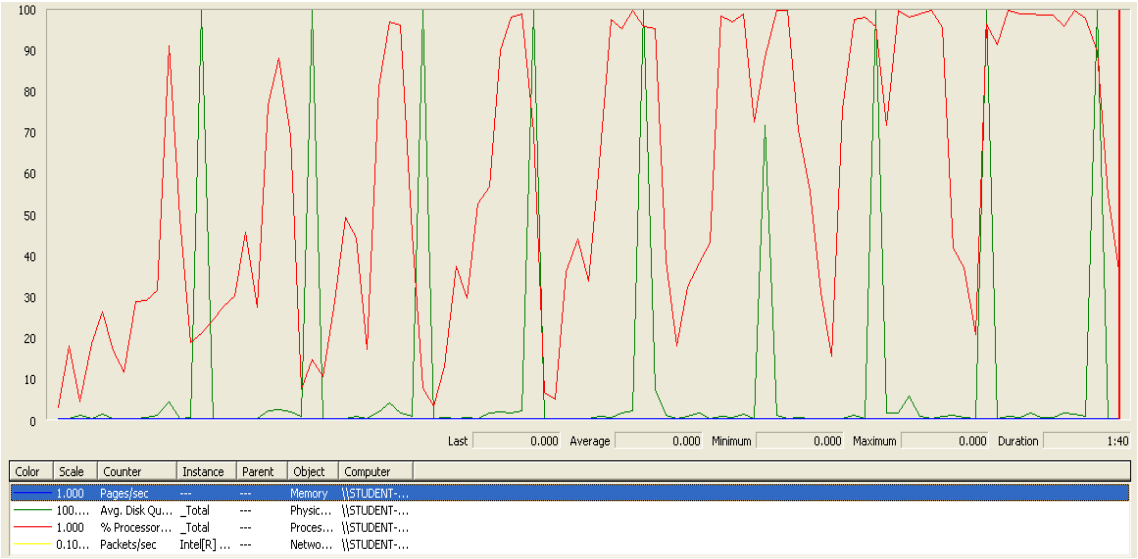


Apparently, there are not errors but the average time per request is not enough good as we can check in the next reports. The system suffers to retrieve the message list of the administrator and when system must send the notification message to all the actors. We have the same problem with the previous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

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	240	104	18	172	9	3070	0.00%	51.6/min	3.3
/j_spring_security_check	240	174	41	355	15	3975	0.00%	51.5/min	3.7
/	480	138	24	210	11	5068	0.00%	1.6/sec	6.3
/message/administrator,company,hacker/list.do	480	1190	833	2610	41	6534	0.00%	1.7/sec	12.1
/message/administrator/broadcast.do	480	983	412	2455	33	10615	0.00%	1.7/sec	10.9
/j_spring_security_logout	240	234	39	413	15	5593	0.00%	50.5/min	3.2
TOTAL	2160	570	103	1794	9	10615	0.00%	7.4/sec	38.5



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.



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.

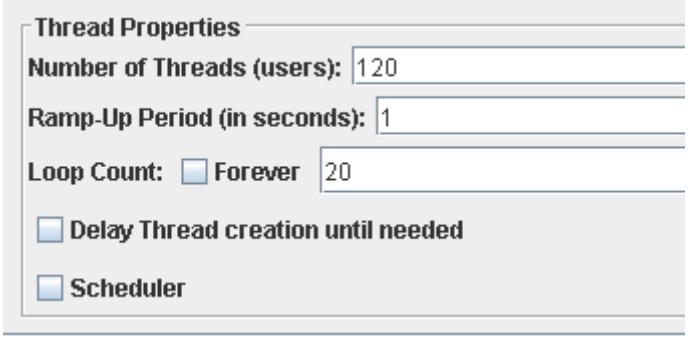
Req. 24.2 - Launch a process that flags the actors of the system as spammers or not-spammers. A user is considered to be a spammer if at least 10% of the messages that he or she's sent contain at least one spam word.

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

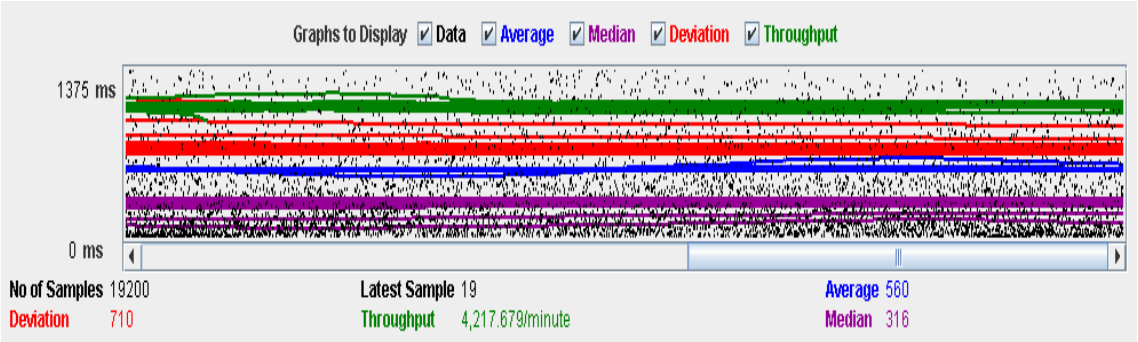
Test case description: first, an administrator logs in to the system. Then, an administrator goes to actor list. Later, an administrator launches the process. Finally, the administrator logs out.

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

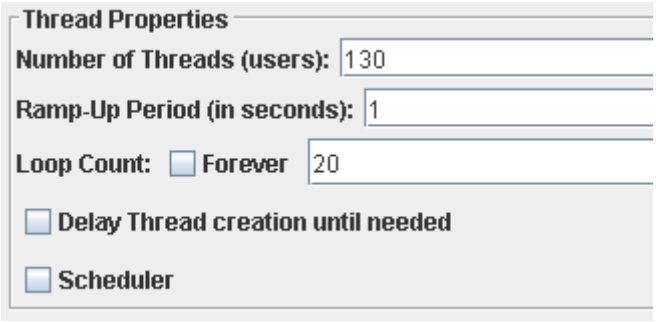


By means of this report, we can appreciate the maximum performance of the system. It doesn't produce errors and the performance is acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	2400	372	162	986	8	4964	0.00%	9.0/sec	35.0
/j_spring_security_check	2400	764	507	1736	17	8437	0.00%	9.0/sec	38.5
/	4800	408	223	1021	9	9965	0.00%	17.7/sec	67.8
/actor/administrator/list.do	4800	498	279	1178	22	7683	0.00%	17.9/sec	103.1
/actor/administrator/spamme...	2400	1128	830	2320	79	7108	0.00%	9.0/sec	54.7
/j_spring_security_logout	2400	405	204	1039	14	7407	0.00%	9.0/sec	34.2
TOTAL	19200	560	316	1367	8	9965	0.00%	70.3/sec	327.8

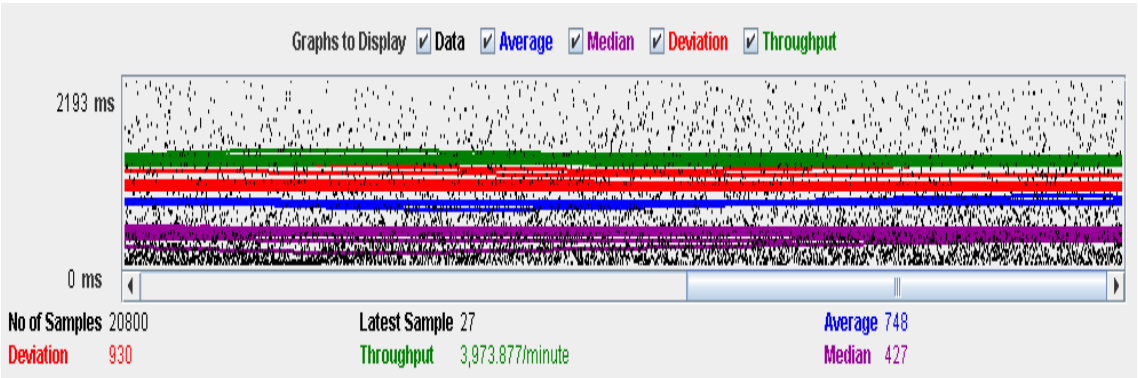


Overload test case: 130 concurrent users, 20 of loop count and 1 as ramp-up period:

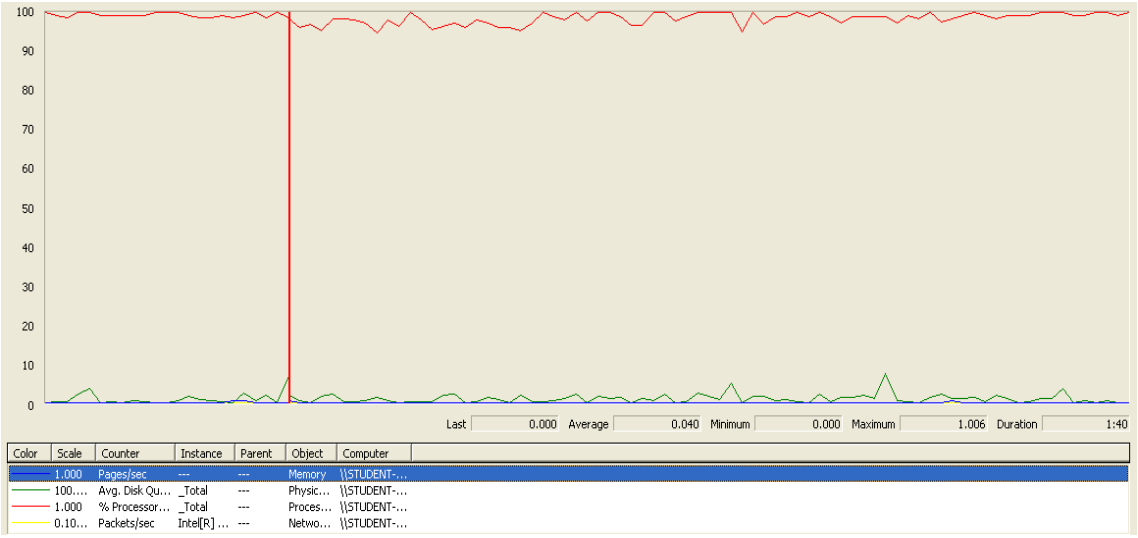


There are not errors but the average time per request is not good. The higher average time corresponds to /actor/administrator/spammer.do. This is an implementation of our application. If we refactor the code, the system would support more concurrent users.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	2600	480	227	1321	6	5688	0.00%	8.5/sec	33.0
/j_spring_security_check	2600	1067	739	2382	19	11573	0.00%	8.5/sec	36.3
/	5200	563	297	1465	11	11281	0.00%	16.6/sec	63.9
/actor/administrator/list.do	5200	671	383	1610	28	13688	0.00%	16.8/sec	96.8
/actor/administrator/spamme...	2600	1429	1056	3046	84	11837	0.00%	8.4/sec	51.4
/j_spring_security_logout	2600	539	293	1340	9	9510	0.00%	8.4/sec	32.1
TOTAL	20800	748	427	1836	6	13688	0.00%	66.2/sec	308.8



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



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

Req. 24.3 – An actor who is authenticated as an administrator must be able to: Ban an actor with the spammer flag.

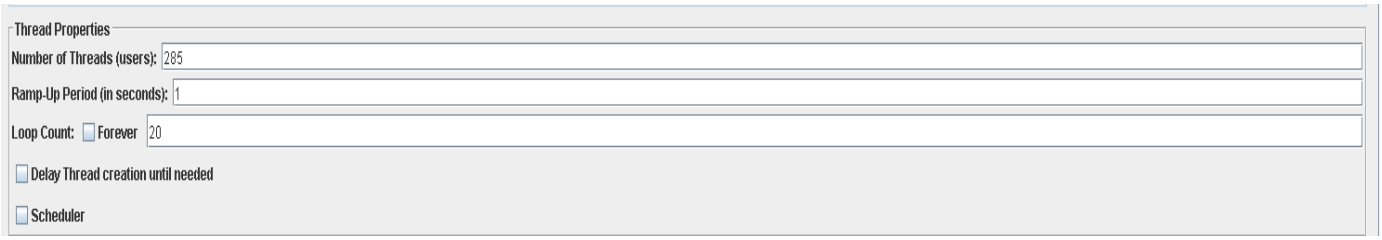
Technical details of the computer on which the test has been executed:

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

Test case description:

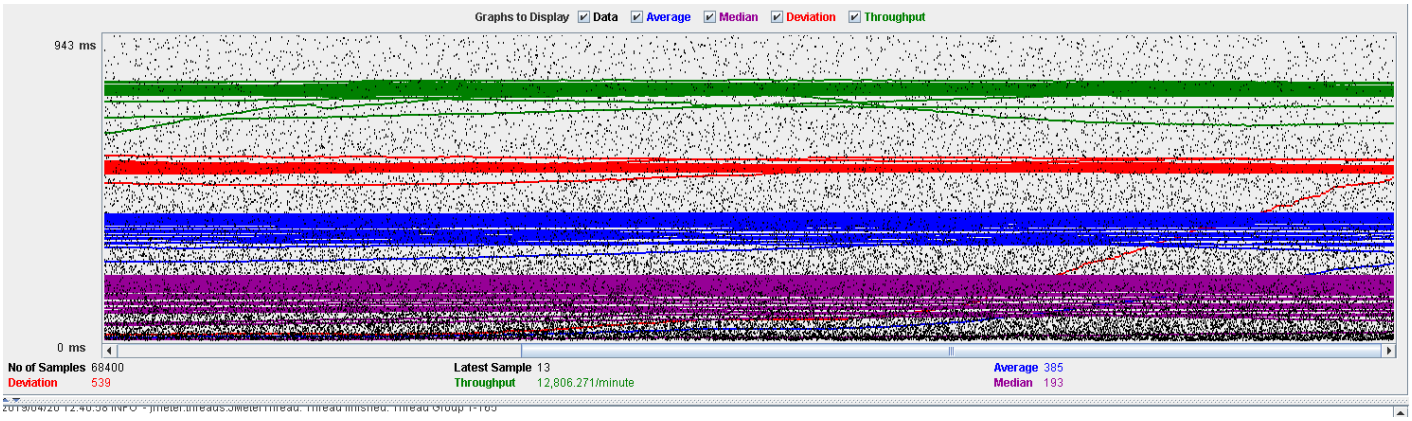
- The user logs in as an administrator.
- The user goes to actor list view.
- The user launchs spammer process.
- The user display the profile of actor who has spammer flag.
- The user bans the actor
- The user logs out the system.

Maximum workload test case. 285 concurrent users and 20 of loop count:



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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	17100	271	124	697	3	6134	0.00%	53.4/sec	202.6
/security/login.do	5700	262	115	670	2	5545	0.00%	18.4/sec	70.3
/j_spring_security_check	5700	542	339	1285	5	7358	0.00%	18.4/sec	79.1
/actor/administrator/list.do	11400	339	171	828	6	8385	0.00%	36.6/sec	207.6
/actor/administrator/spa...	5700	732	490	1656	17	7174	0.00%	18.4/sec	110.1
/actor/display.do	11400	325	166	826	6	5911	0.00%	36.6/sec	186.6
/actor/administrator/ban....	5700	635	428	1415	12	6307	0.00%	18.4/sec	98.2
/j_spring_security_logout	5700	305	148	768	4	5678	0.00%	18.4/sec	70.7
TOTAL	68400	385	193	970	2	8385	0.00%	213.4/sec	1000.4

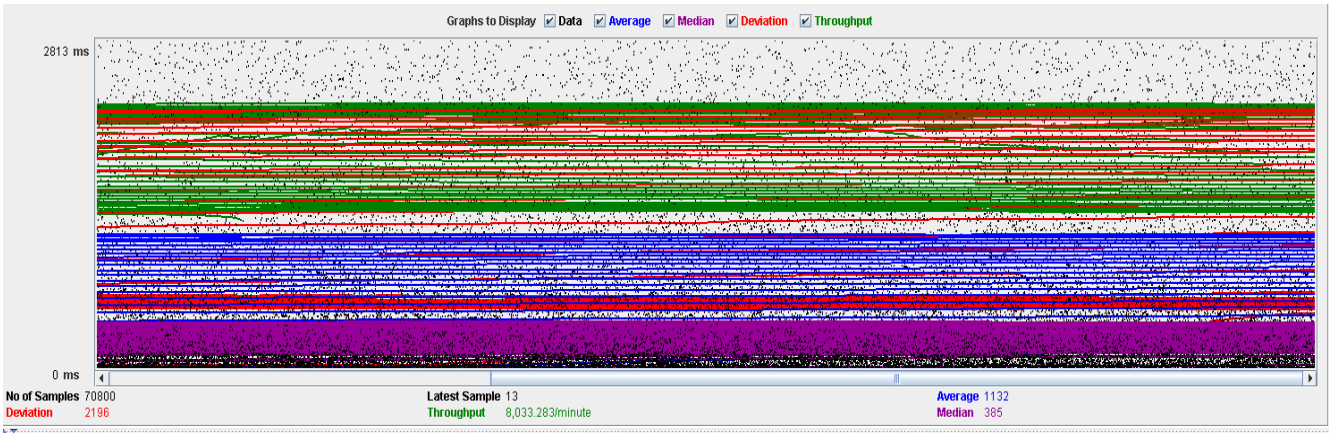


Overload test case: 295 concurrent users and 20 of loop count:

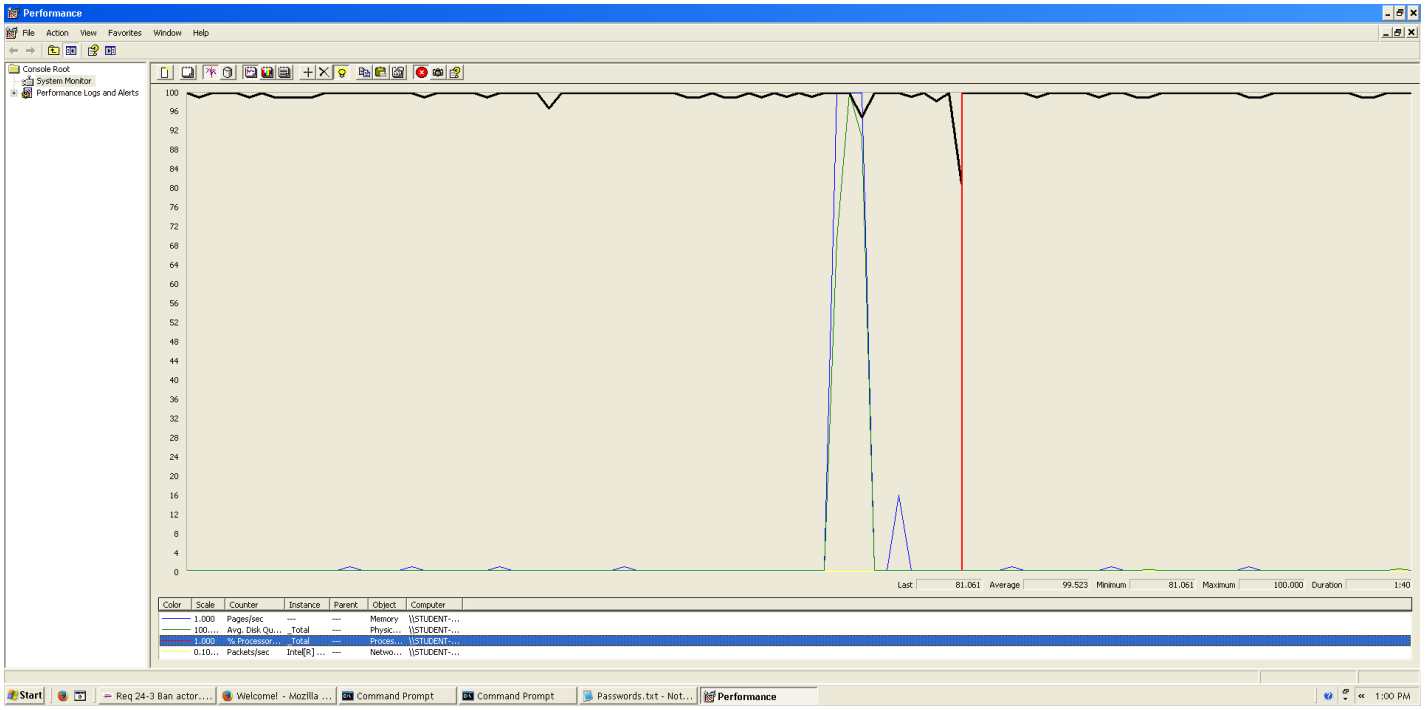
We don't have any errors, but the system works with excessive delay.

As we can see in the table below, the request with the highest average time is /actor/administrator/spammerProcess. This URI correspond to a method that goes over all the actors of the system and analyze their messages to set spammer flags so we can improve and refactoring the code to increase the users.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	17700	856	244	2250	3	38387	0.00%	33.5/sec	127.2
/security/login.do	5900	847	201	2129	2	49893	0.00%	11.4/sec	43.7
/j_spring_security_check	5900	1707	717	4378	5	38854	0.00%	11.4/sec	49.2
/actor/administrator/list.do	11800	927	322	2261	7	45039	0.00%	22.7/sec	129.1
/actor/administrator/spa...	5900	1933	999	4582	17	51272	0.00%	11.4/sec	68.3
/actor/display.do	11800	950	336	2334	5	35185	0.00%	22.7/sec	115.9
/actor/administrator/ban....	5900	1825	915	4432	11	40039	0.00%	11.4/sec	60.9
/j_spring_security_logout	5900	946	281	2465	4	40640	0.00%	11.4/sec	43.8
TOTAL	70800	1132	385	2903	2	51272	0.00%	133.9/sec	628.1



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.



Conclusion: The maximum number of concurrent users supported by this test case is 285 and we could improve it by assigning more CPU's resources to our system and refactoring our code to try to increase the number of concurrent users.

Req. 24.4 – An actor who is authenticated as an administrator must be able to: Unban an actor with the spammer flag.

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

- Ram: 8,00 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

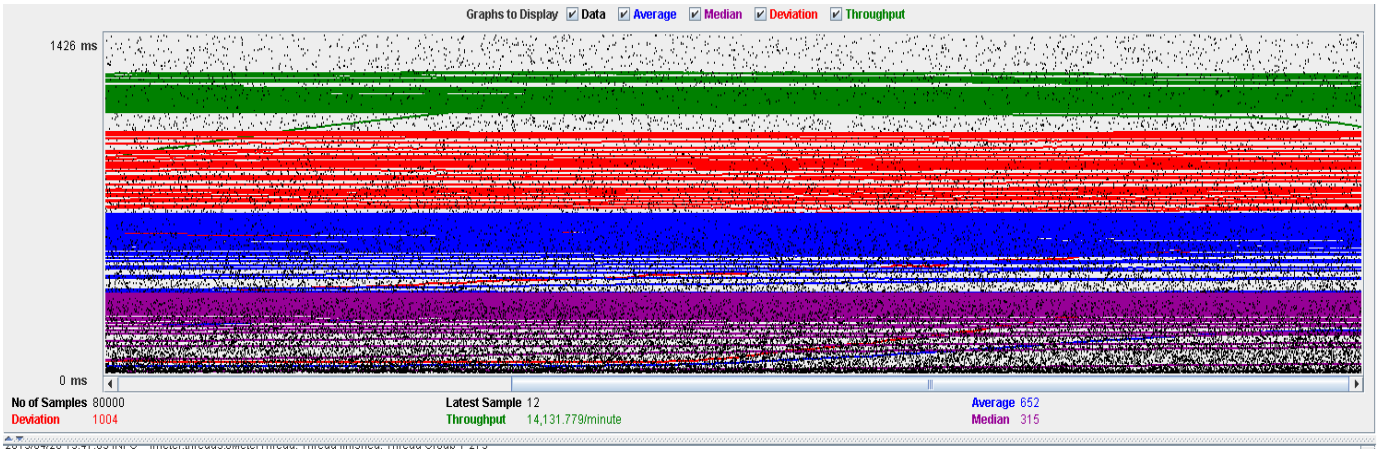
Test case description:

- The user logs in as an administrator.
- The user goes to actor list view.
- The user display the profile of actor who has banned previously.
- The user unbans the actor
- The user logs out the system.

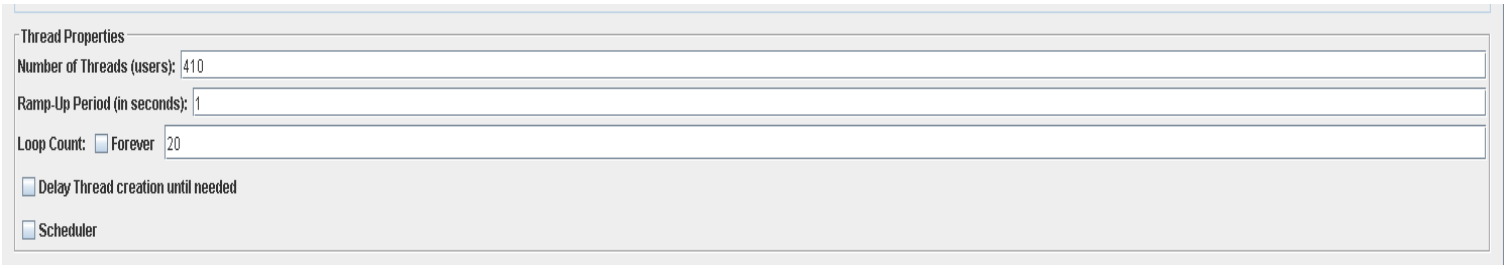
Maximum workload test case. 400 concurrent users and 20 of loop count:

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.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	24000	537	249	1319	3	19482	0.00%	70.7/sec	268.7
/security/login.do	8000	526	232	1328	3	13202	0.00%	24.2/sec	92.6
/j_spring_security_check	8000	1062	662	2455	5	18664	0.00%	24.2/sec	104.5
/actor/administrator/list.do	8000	557	261	1355	8	17796	0.00%	24.2/sec	135.7
/actor/display.do	16000	556	263	1366	5	16119	0.00%	48.1/sec	245.7
/actor/administrator/unB...	8000	1104	711	2471	8	17655	0.00%	24.2/sec	103.0
/j_spring_security_logout	8000	545	264	1348	4	12862	0.00%	24.2/sec	93.0
TOTAL	80000	652	315	1606	3	19482	0.00%	235.5/sec	1023.9

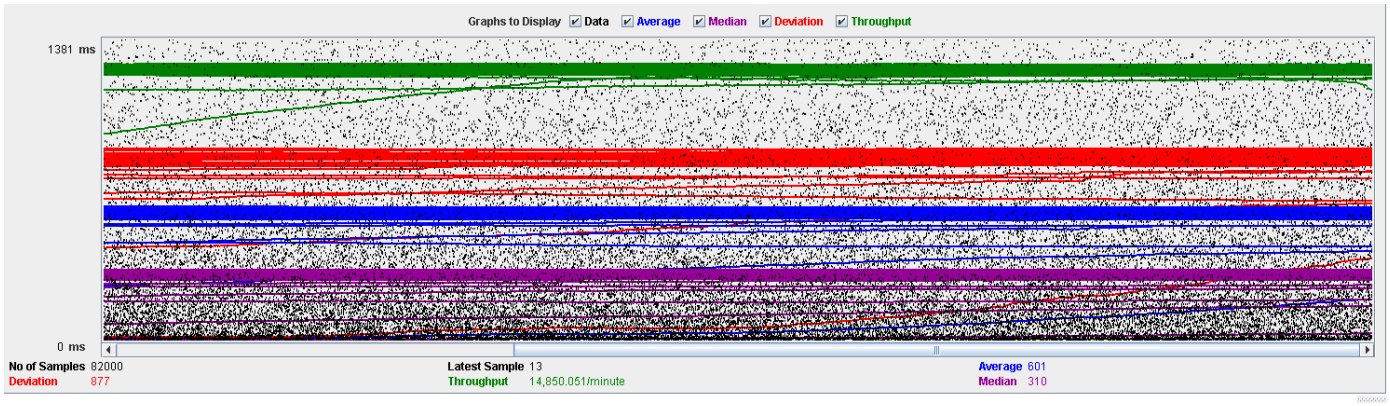


Overload test case: 410 concurrent users and 20 of loop count:

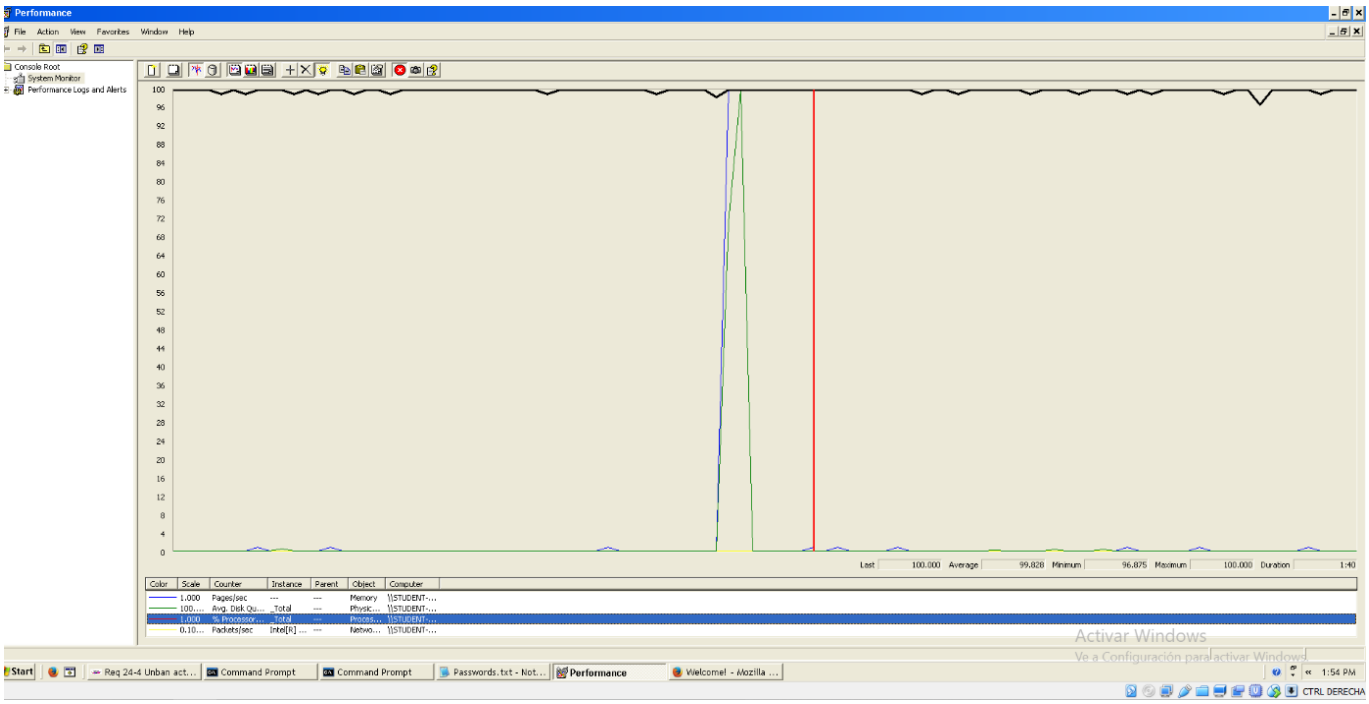


Even though the average time per request is still acceptable, it begins to produce some errors as we can see in the following picture. The errors are always the same: I/O exception (java.net.SocketException) caught when processing request: Connection reset by peer: socket write error This exception is not related with the implementation of our application, but with tomcat. The system can't handle this number of concurrent users properly.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
	24600	490	243	1193	0	22435	0.21%	74.3/sec	282.0
ecurity/login.do	8200	477	225	1189	1	22046	0.23%	25.5/sec	97.4
_spring_security_check	8200	960	639	2154	1	13761	0.06%	25.5/sec	109.9
ctor/administrator/list.do	8200	516	268	1272	1	9558	0.04%	25.5/sec	142.7
ctor/display.do	16400	511	259	1246	1	20436	0.20%	50.6/sec	258.0
ctor/administrator/unB...	8200	1036	683	2315	0	11805	0.20%	25.4/sec	108.2
_spring_security_logout	8200	526	270	1280	1	11724	0.22%	25.4/sec	97.7
OTAL	82000	601	310	1464	0	22435	0.18%	247.5/sec	1074.9



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.



Conclusion: The maximum number of concurrent users supported by this test case is 400 and we could improve it by assigning more CPU's resources to our system.

CONCLUSIÓN

As we can see the system offers an acceptable performance that could be improved if we increase the resources of the machines.

We can verify that not all use cases admit the same number of users concurrently. They can range from more than 400 to 10 in case of handling the messages. The latter being the maximum number of concurrent users that our system supports. Although it seems a low number, it must be said that this use case is not widely used.