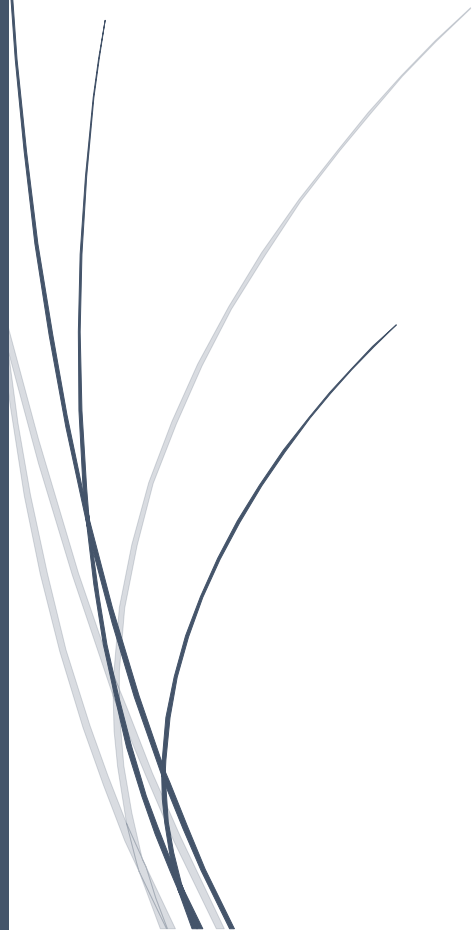
16-5-2019

Acme Personal Trainer

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.

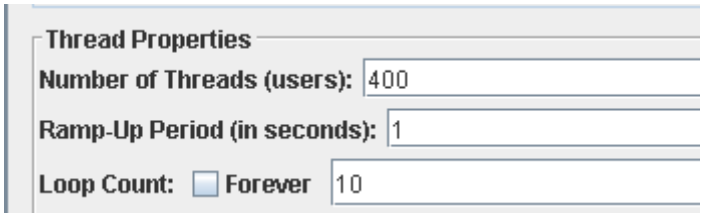
PERFORMANCE TESTING

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

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

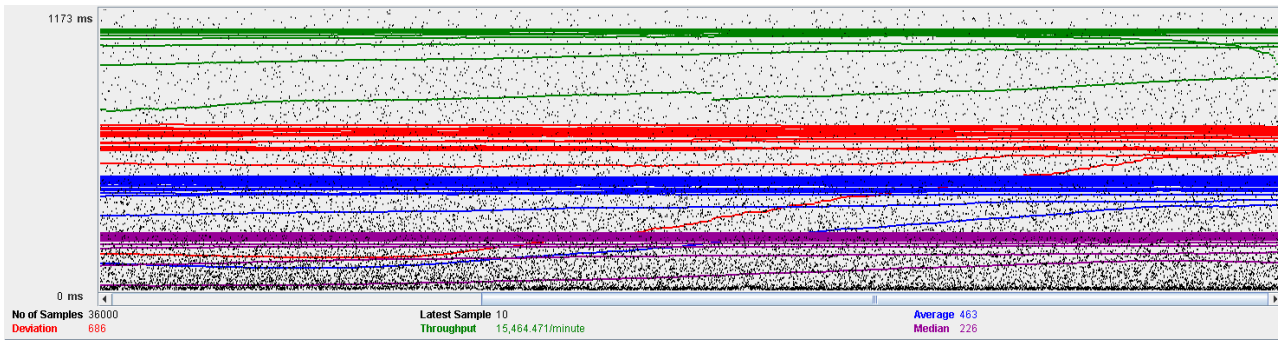
Test case description: The user goes to *Register customer* option menu, he/she fills the form and save it. Then, the user logs in the system and finally, logs out.

Maximum workload test case: 400 concurrent users, 10 of loop count and 1 of ramp-up period.

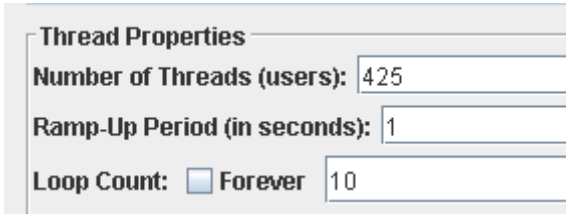


According to the performance test, this is the maximum workload that can be supported by the system without errors and failures or insufficient performance. We can see a good response time with this users amount.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	12000	405	192	1030	3	12383	0.00%	85.9/sec	318.7
/actor/registerCustomer....	8000	437	218	1108	4	7635	0.00%	60.0/sec	439.7
/welcome/index.do	4000	424	216	1084	3	9798	0.00%	30.3/sec	101.9
/security/login.do	4000	427	195	1059	3	8513	0.00%	30.3/sec	108.1
/j_spring_security_check	4000	798	495	1844	5	10521	0.00%	30.3/sec	138.0
/j_spring_security_logout	4000	424	199	1105	5	8122	0.00%	30.4/sec	109.0
TOTAL	36000	463	226	1172	3	12383	0.00%	257.7/sec	1169.9



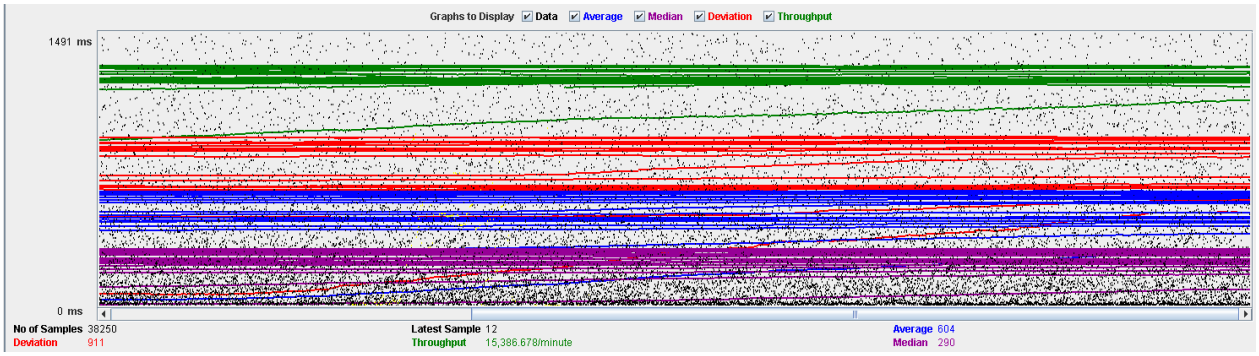
Overload test case: 425 concurrent users, 10 of loop count and 1 as ramp-up period.



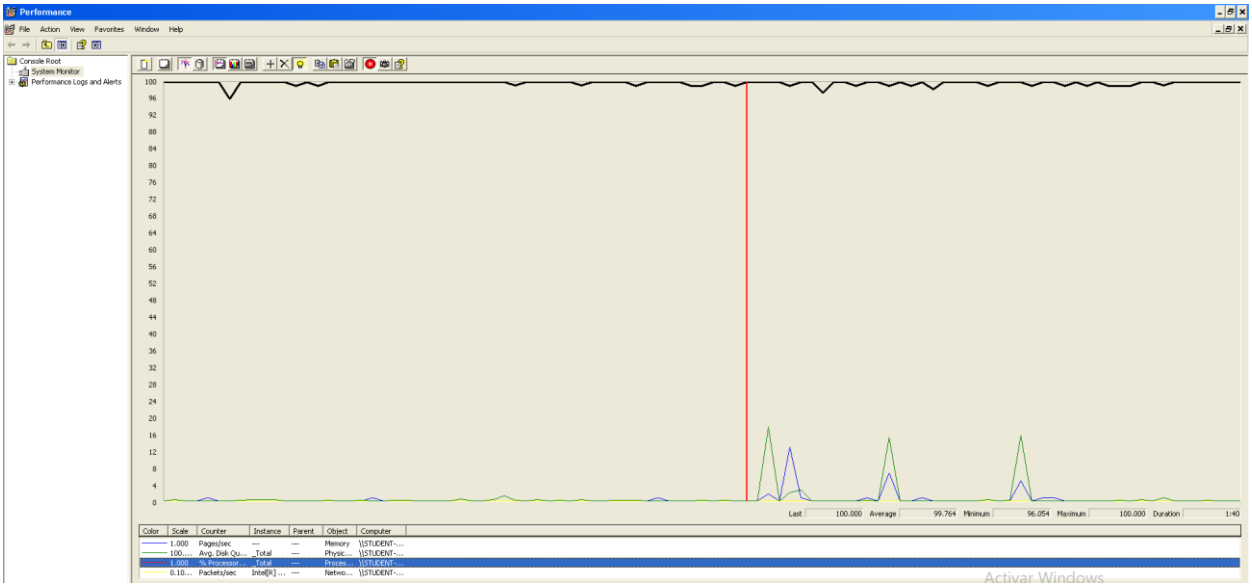
Iit begins to produce some errors as we can see in the following picture. The errors are always the same: I/O exception (java.net.SocketException) caught when processing request: Connection reset by peer: socket write error.

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
/	12750	518	240	1314	1	11381	0.37%	85.5/sec	316.6
/actor/registerCustomer....	8500	578	269	1448	4	15420	0.52%	59.4/sec	433.4
/welcome/index.do	4250	557	267	1406	1	11129	0.66%	30.0/sec	100.6
/security/login.do	4250	545	250	1315	2	13892	0.35%	30.0/sec	106.9
/j_spring_security_check	4250	1066	671	2535	6	18090	0.49%	30.0/sec	136.4
/j_spring_security_logout	4250	556	266	1391	2	10257	0.56%	30.0/sec	107.3
TOTAL	38250	604	290	1512	1	18090	0.47%	256.4/sec	1161.2



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.



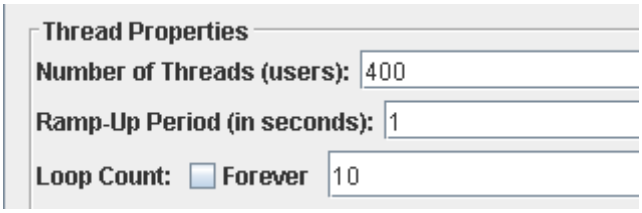
Conclusion: The maximum number of concurrent users supported by the system is 400 because the current CPU is a bottleneck.

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

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

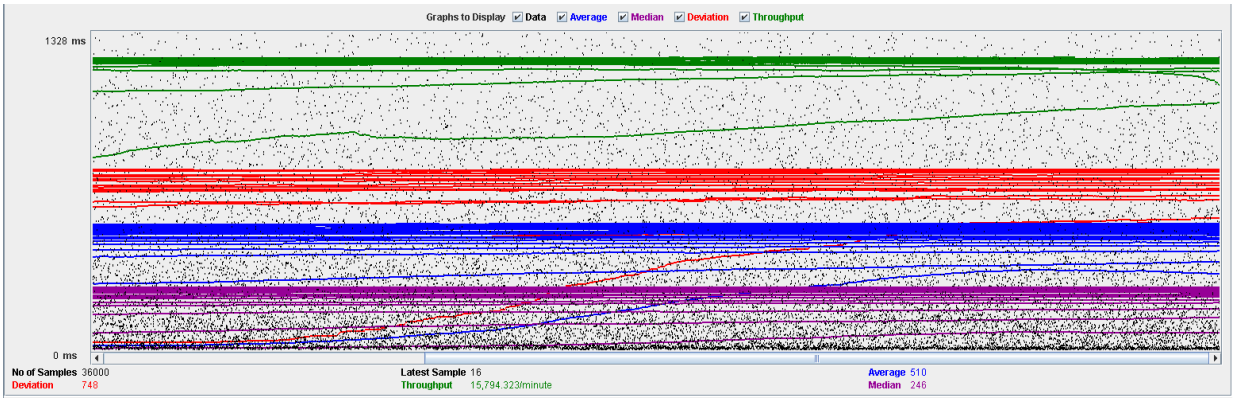
Test case description: The user goes to *Register trainer* option menu, he/she fills the form and save it. Then, the user logs in the system and finally, logs out.

Maximum workload test case: 400 concurrent users, 10 of loop count and 1 of ramp-up period.

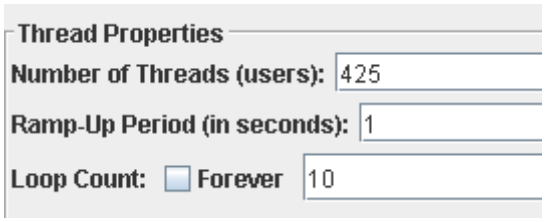


According to the performance test, this is the maximum workload that can be supported by the system without errors and failures or insufficient performance. The response times are really good for this users amount.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	12000	448	203	1165	2	8483	0.00%	87.7/sec	309
/actor/registerTrainer.do	8000	471	231	1213	4	9854	0.00%	61.1/sec	447
/welcome/index.do	4000	463	233	1175	4	9714	0.00%	30.9/sec	104
/security/login.do	4000	447	212	1118	3	7384	0.00%	30.9/sec	110
/j_spring_security_check	4000	920	595	2123	8	11005	0.00%	30.9/sec	124
/j_spring_security_logout	4000	477	224	1187	3	10222	0.00%	31.1/sec	111
TOTAL	36000	510	246	1308	2	11005	0.00%	263.2/sec	1163

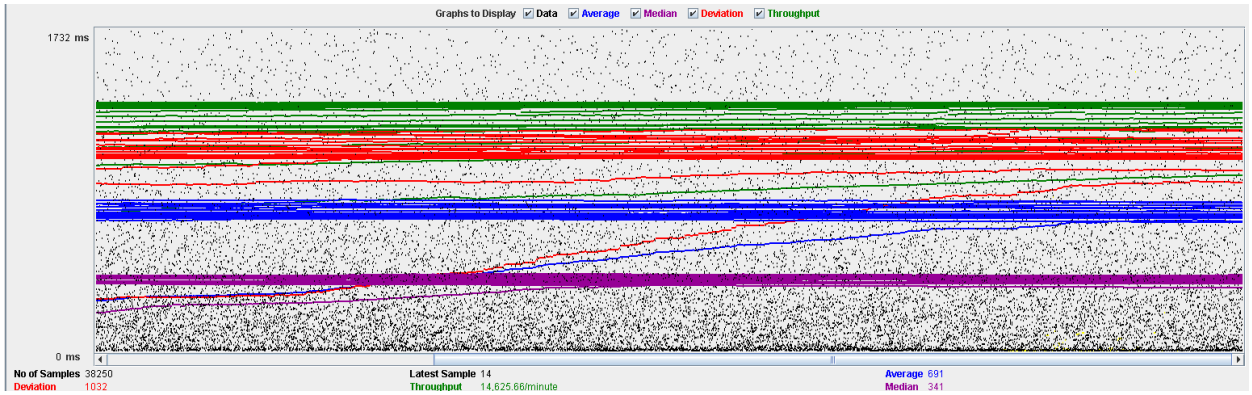


Overload test case: 425 concurrent users, 10 of loop count and 1 as ramp-up period.

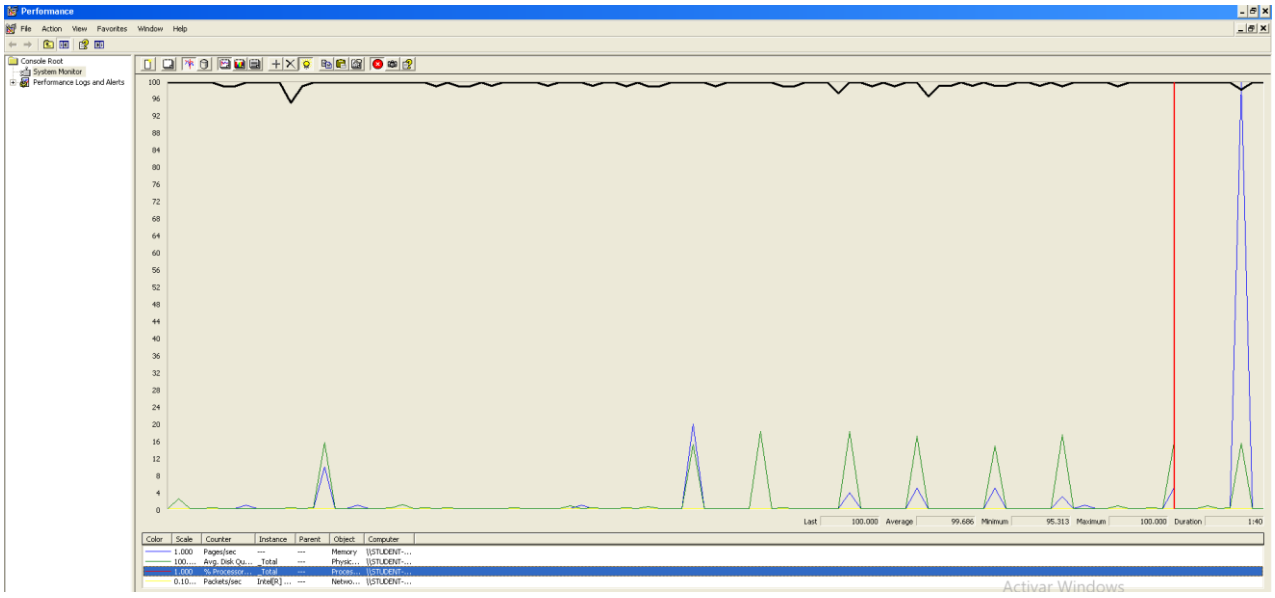


The requests with highest time is “j_spring_security_check” this is an internal unit of Spring. That’s means that we cannot improve the performance by refactoring the code. There are some errors too.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	12750	604	285	1501	0	14081	0.42%	81.3/sec	286.3
/actor/registerTrainer.do	8500	673	331	1671	0	14310	0.46%	56.4/sec	411.6
/welcome/index.do	4250	608	295	1541	0	12231	0.52%	28.5/sec	95.7
/security/login.do	4250	629	310	1618	0	13226	0.31%	28.6/sec	102.0
/j_spring_security_check	4250	1210	778	2686	1	15031	0.59%	28.6/sec	114.9
/j_spring_security_logout	4250	620	315	1499	0	13901	0.31%	28.9/sec	103.5
TOTAL	38250	691	341	1737	0	15031	0.43%	243.8/sec	1074.6



In the following performance analysis we can appreciate that CPU is running at 95-100% all the time. So, there is a bottleneck with CPU. If we use a more powerful CPU, the maximum workload would be higher.



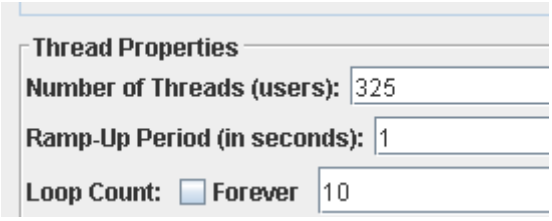
Conclusion: The maximum number of concurrent users supported by the system is 400 because the current CPU is a bottleneck.

Req. 8.2 - An actor who is authenticated must be able to: Edit his or her personal data.

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

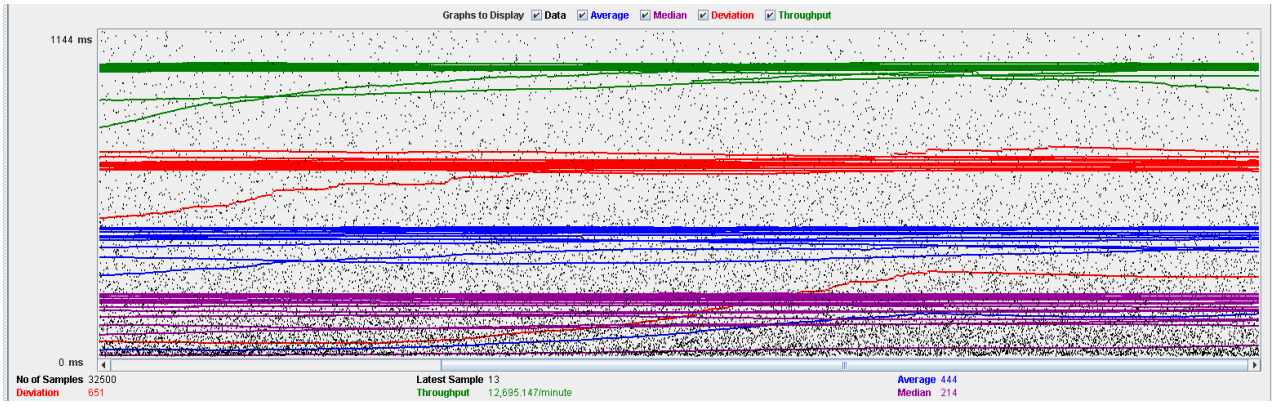
Test case description: For this case, we have analysed one of the actors of the system, it would be customer. The customer logs in the system and select *Display profile* in the main menu, next, he/she clicks on *Edit Personal Data* and changes the fields that want to update.

Maximum workload test case: 325 concurrent users, 10 of loop count and 1 of ramp-up period.

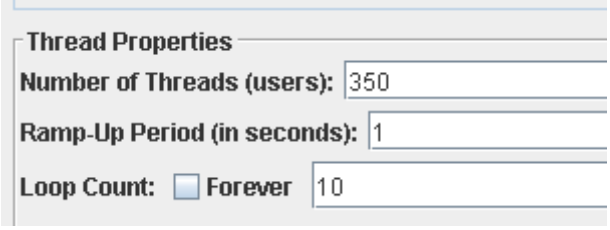


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	9750	337	159	847	2	5804	0.00%	63.5/sec	235.3
/security/login.do	3250	321	132	843	3	7398	0.00%	22.5/sec	80.3
/j_spring_security_check	3250	610	376	1380	7	7849	0.00%	22.5/sec	102.3
/actor/display.do	6500	405	191	1002	6	7197	0.00%	44.0/sec	247.3
/actor/administrator,audit...	6500	646	348	1592	5	7445	0.00%	44.4/sec	302.7
/j_spring_security_logout	3250	395	197	989	5	5361	0.00%	22.5/sec	80.5
TOTAL	32500	444	214	1124	2	7849	0.00%	211.6/sec	1009.2

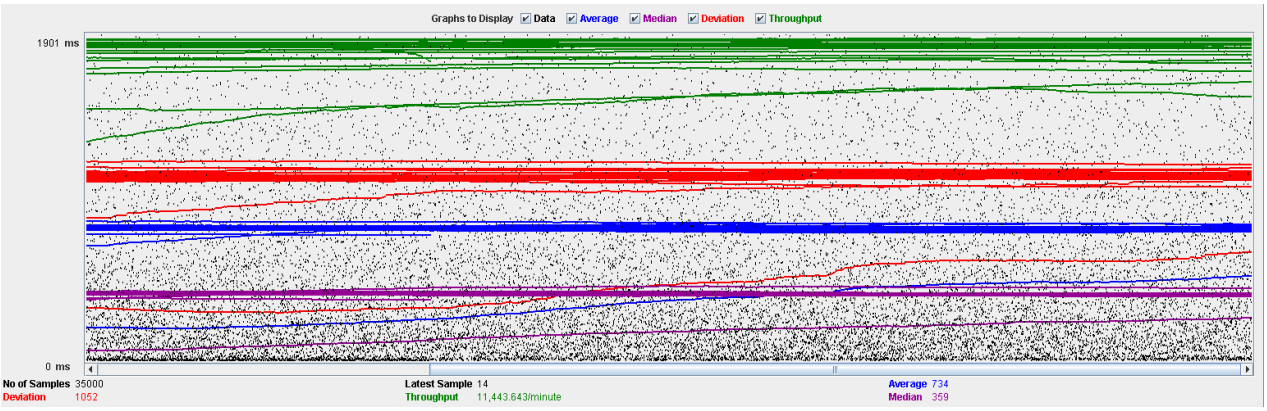


Overload test case: 350 concurrent users, 10 of loop count and 1 as ramp-up period.

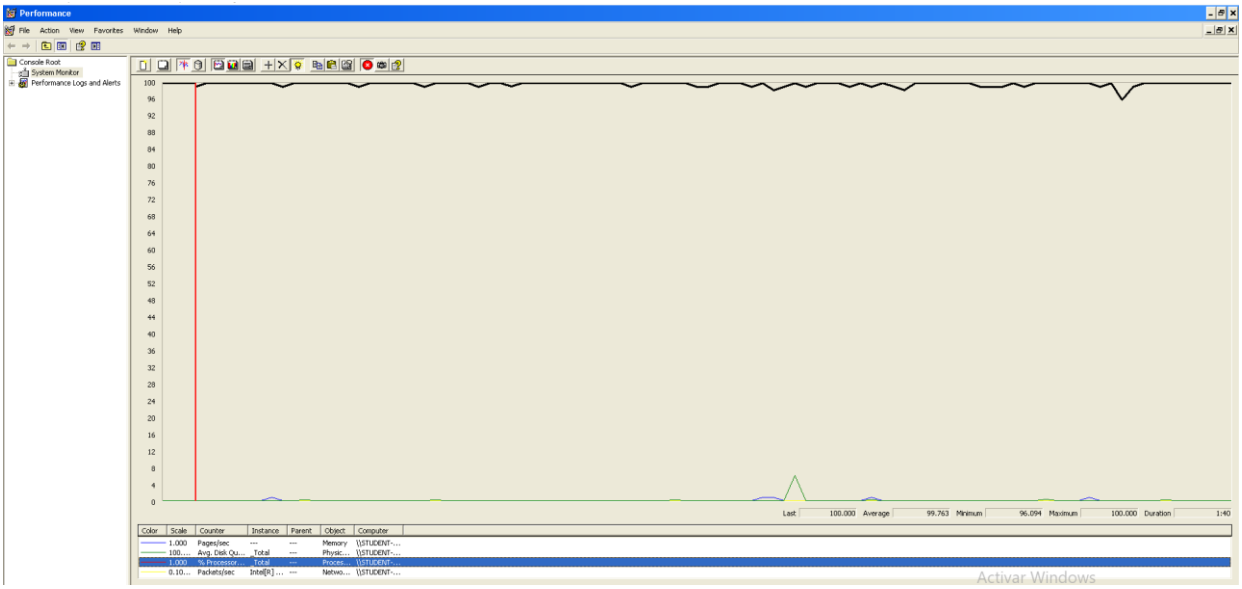


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	10500	580	271	1454	4	16537	0.00%	57.2/sec	212.2
/security/login.do	3500	554	257	1402	3	11944	0.00%	20.1/sec	71.8
/j_spring_security_check	3500	1188	750	2833	8	11749	0.00%	20.1/sec	91.7
/actor/display.do	7000	626	299	1566	7	9088	0.00%	39.5/sec	222.4
/actor/administrator,audit...	7000	992	533	2484	7	17203	0.00%	39.9/sec	271.8
/j_spring_security_logout	3500	622	312	1586	6	15533	0.00%	20.3/sec	72.9
TOTAL	35000	734	359	1886	3	17203	0.00%	190.7/sec	910.4



With the following picture is performance analysis in which we can see that CPU are working almost 100%. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.



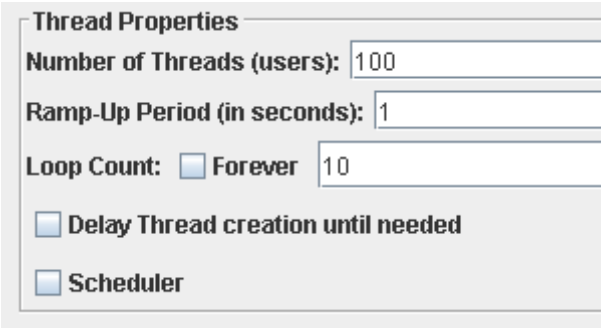
Conclusion: The maximum number of concurrent users supported by the system is 325 because the current CPU is a bottleneck.

Req. 8.3 - An actor who is authenticated must be able to exchange messages with other actors and manage them, which includes listing, showing, sending and deleting them.

- 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 + 1TB GB HDD
- Network card: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter.

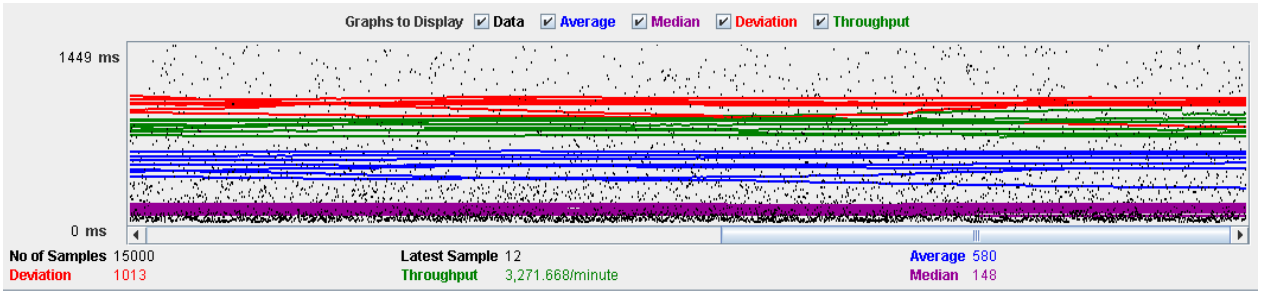
Test case description: First, the user login as a trainer ("trainer1"). Then he or she goes to boxes list and clicks "Send a message" link. Later, the user fills the form and press send button. Below, the user displays out box to see the sent messages list. And now, he or she displays the last sent message. The user return to out box and clicks delete link regarding the last sent message. Eventually, the user logs out.

Maximum workload test case: 100 concurrent users, 10 of loop count and 1 of ramp-up period.

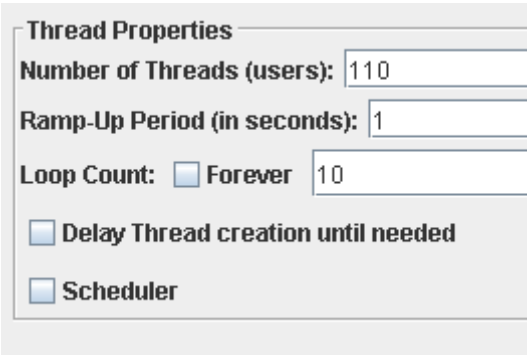


With those parameters, the system has a suitable behaviour: there aren't any errors or failures and response time is good.

Label	# Sampl...	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	3000	365	72	1023	9	9293	0.00%	10.9/sec	38.3
/security/login.do	1000	337	44	888	8	8987	0.00%	4.0/sec	14.1
/j_spring_security_check	1000	818	246	2319	19	8151	0.00%	3.9/sec	15.7
/box/administrator,auditor,customer,nutritionist,trainer/list.do	3000	546	124	1624	15	9002	0.00%	11.1/sec	55.6
/message/administrator,auditor,customer,nutritionist,trainer/send.do	2000	977	536	2461	40	10386	0.00%	7.7/sec	44.6
/box/administrator,auditor,customer,nutritionist,trainer/display.do	2000	456	124	1302	25	7586	0.00%	7.6/sec	61.7
/message/administrator,auditor,customer,nutritionist,trainer/display.do	1000	837	359	2223	21	12189	0.00%	3.9/sec	15.3
/message/administrator,auditor,customer,nutritionist,trainer/delete.do	1000	759	277	2228	27	9303	0.00%	3.9/sec	15.2
/j_spring_security_logout	1000	345	79	974	13	9427	0.00%	3.9/sec	13.8
TOTAL	15000	580	148	1680	8	12189	0.00%	54.5/sec	262.5

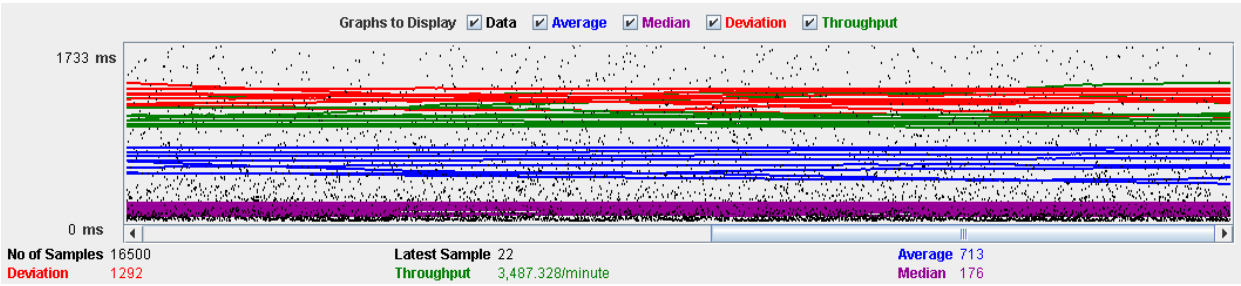


Overload test case: 110 concurrent users, 10 of loop count and 1 as ramp-up period.



In spite of lack of errors, the average time per request is not acceptable. The requests with highest average time are “j_spring_security_check” and send a message. On the one hand, “j_spring_security_check” is an internal unit of Spring. That’s means that we cannot improve the performance by refactoring the code. On the other hand, we can improve the performance of send a message. Every time a user sends a message, the system must check if this message contains any spam word. This verification is inefficient.

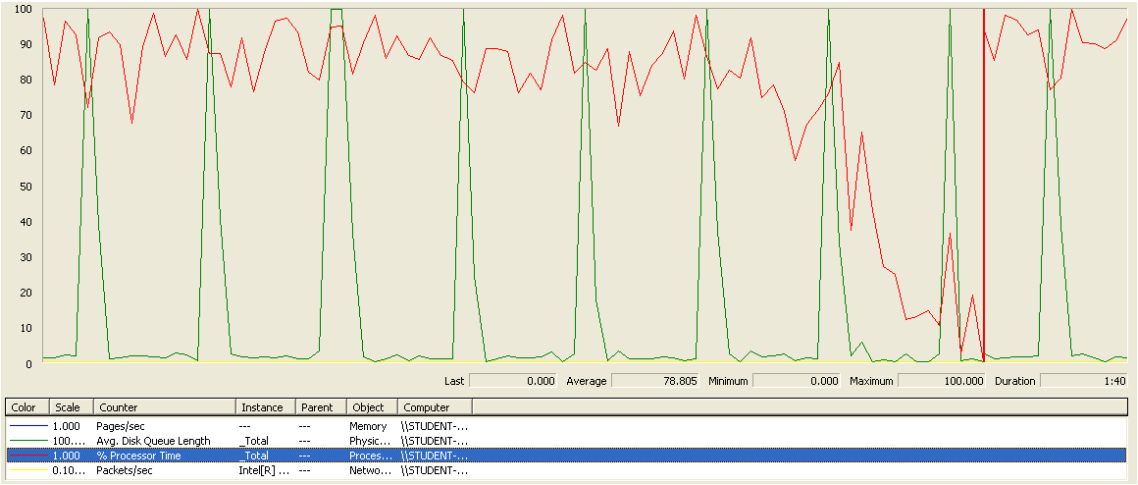
Label	# Sampl...	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	3300	439	62	1176	9	15246	0.00%	11.6/sec	40.8
/security/login.do	1100	445	56	1239	8	11241	0.00%	4.1/sec	14.5
/j_spring_security_check	1100	981	280	2950	19	11275	0.00%	4.1/sec	16.3
/box/administrator,auditor,customer,nutritionist,trainer/list.do	3300	631	131	1822	10	11764	0.00%	11.8/sec	59.1
/message/administrator,auditor,customer,nutritionist,trainer/send.do	2200	1184	649	2978	33	13392	0.00%	8.1/sec	46.9
/box/administrator,auditor,customer,nutritionist,trainer/display.do	2200	598	157	1673	22	12104	0.00%	8.1/sec	65.3
/message/administrator,auditor,customer,nutritionist,trainer/display.do	1100	1027	445	2774	19	11955	0.00%	4.1/sec	16.1
/message/administrator,auditor,customer,nutritionist,trainer/delete.do	1100	921	434	2271	25	10551	0.00%	4.1/sec	16.1
/j_spring_security_logout	1100	544	105	1505	13	12135	0.00%	4.1/sec	14.6
TOTAL	16500	713	176	1993	8	15246	0.00%	58.1/sec	279.9



In the following performance analysis, we can appreciate that there are peaks at 100% related with hard disk. Besides, the CPU component works at 70-100 %.

Hard disk has peaks when the system retrieves messages list of a box from the database. It seems that hard disk is not powerful enough.

Regarding CPU, we could get a better performance improving it but not too much because, as we can see in the graphic, the CPU is not always working at 90-100%.



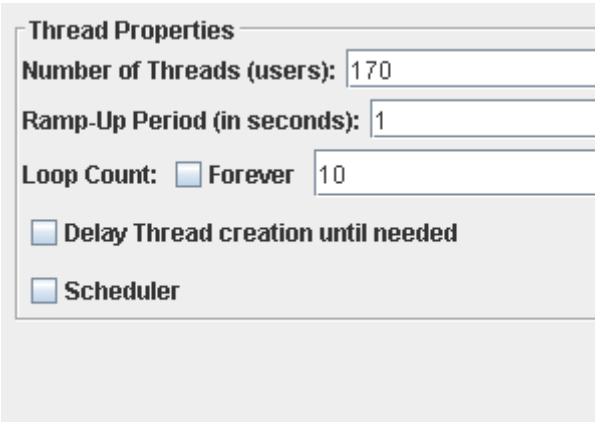
Conclusion: The maximum number of concurrent users supported by the system is 100. The maximum workload would be higher if we can get refactor the code and replace the hard diks and CPU.

Req. 8.4 - An actor who is authenticated must be able to manage his or her message boxes, which includes listing, showing, creating, updating, moving and deleting them. Except for the system boxes which only can be listed and showed.

- 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 + 1TB GB HDD
- Network card: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter.

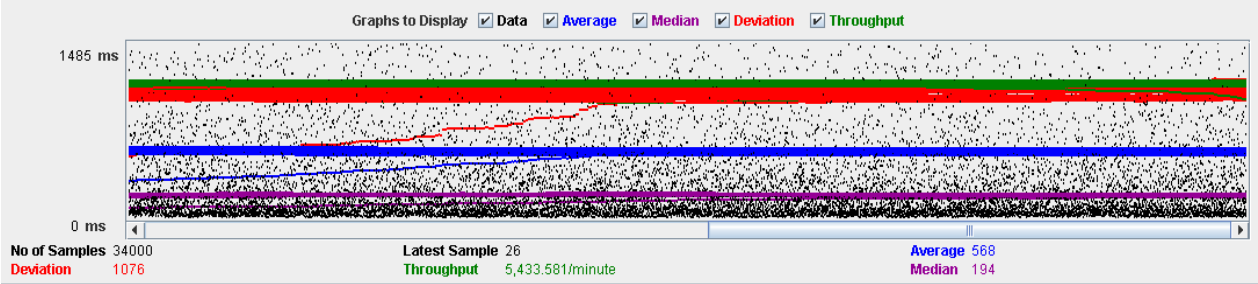
Test case description: First, the user login as a trainer ("trainer1"). Then he or she goes to boxes list and clicks "Create a box" link. Later, the user fills the form and press save button. Below, the user displays the newly created box. And now, he or she edits the box that was created previously. Next, the user deletes the edited box. Eventually, the user logs out.

Maximum workload test case: 170 concurrent users, 10 of loop count and 1 of ramp-up period.

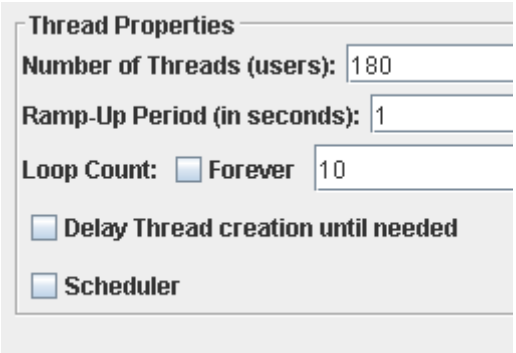


This is the maximum workload that can be supported by the system. The system doesn't produce any errors or failures and the response time is correct.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	5100	478	150	1180	11	14062	0.00%	13.6/sec	47.8
/security/login.do	1700	438	125	962	9	12643	0.00%	4.9/sec	17.5
/j_spring_security_check	1700	975	458	2420	20	14638	0.00%	4.9/sec	19.7
/box/administrator,auditor,customer,nutritionist,trainer/list.do	11900	500	169	1208	14	15581	0.00%	32.4/sec	175.1
/box/administrator,auditor,customer,nutritionist,trainer/create.do	1700	558	186	1283	22	11970	0.00%	4.9/sec	24.1
/box/administrator,auditor,customer,nutritionist,trainer/edit.do	8500	701	265	1758	15	14063	0.00%	23.3/sec	169.5
/box/administrator,auditor,customer,nutritionist,trainer/display.do	1700	469	159	1064	19	13495	0.00%	4.9/sec	18.6
/j_spring_security_logout	1700	482	143	1227	14	11564	0.00%	4.9/sec	17.3
TOTAL	34000	568	194	1387	9	15581	0.00%	90.6/sec	473.3

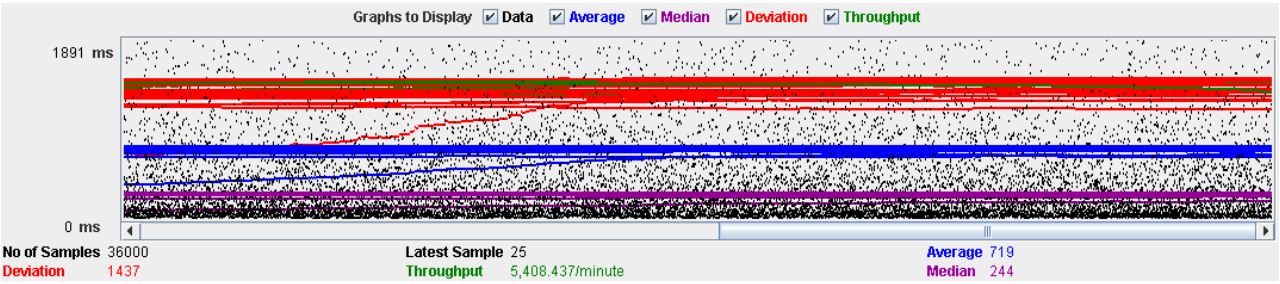


Overload test case: 180 concurrent users, 10 of loop count and 1 as ramp-up period.

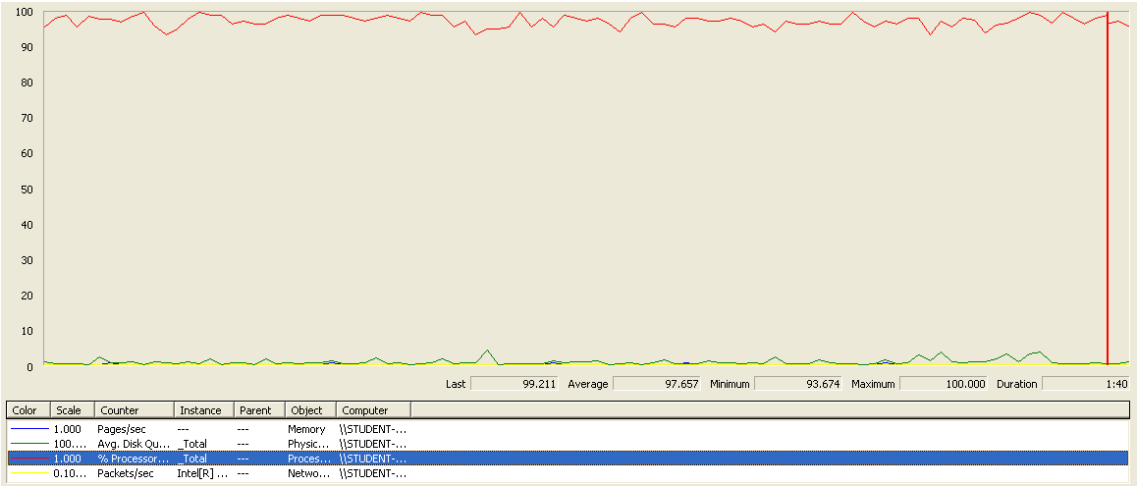


The guilty that system cannot support a higher workload is the unit known as "j_spring_security_check". This is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	5400	614	177	1443	10	20431	0.00%	13.5/sec	47.6
/security/login.do	1800	563	145	1400	9	23391	0.00%	4.8/sec	17.0
/j_spring_security_check	1800	1314	565	3271	19	16875	0.00%	4.8/sec	19.1
/box/administrator,auditor,customer,nutritionist,trainer/list.do	12600	639	215	1479	13	25500	0.00%	32.0/sec	167.6
/box/administrator,auditor,customer,nutritionist,trainer/create.do	1800	677	236	1634	20	15145	0.00%	4.8/sec	23.4
/box/administrator,auditor,customer,nutritionist,trainer/edit.do	9000	851	322	2055	17	18587	0.00%	23.1/sec	167.5
/box/administrator,auditor,customer,nutritionist,trainer/display.do	1800	654	224	1638	19	12374	0.00%	4.8/sec	18.2
/j_spring_security_logout	1800	597	176	1443	10	14838	0.00%	4.8/sec	17.2
TOTAL	36000	719	244	1708	9	25500	0.00%	90.1/sec	465.4



It’s obvious that there is a bottleneck in the CPU. It is saturated (CPU is working at 90%-100% all the time).



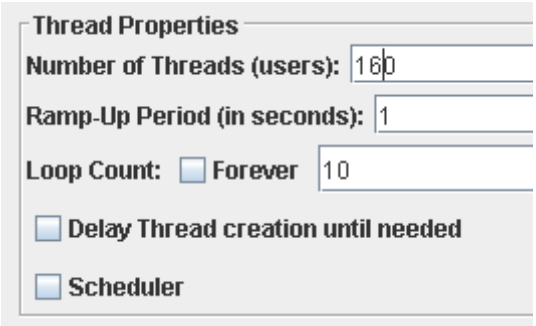
Conclusion: The maximum number of concurrent users supported by the system is 170. There is a bottleneck in CPU. Replacing it by a more powerful component might boost the system. It’s not necessary to update the other components.

Req. 8.5 - An actor who is authenticated must be able to manage his or her social profiles, which includes listing, showing, creating, updating and deleting them.

- 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 + 1TB GB HDD
- Network card: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter.

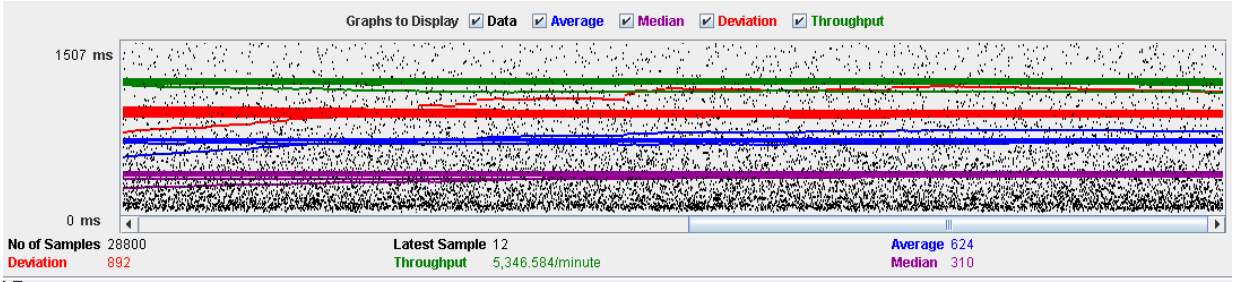
Test case description: First, the user login as a trainer (“trainer1”). Then he or she goes to his or her profile and clicks “Social profiles” link. Later, the user clicks “create a new social profile” link. Immediately, the user fills the form and press save button. Below, he or she edits the social profile. Now, a social profile is displayed. Straightaway, the user deletes the edited social profile. Eventually, the user logs out.

Maximum workload test case: 160 concurrent users, 10 of loop count and 1 of ramp-up period.

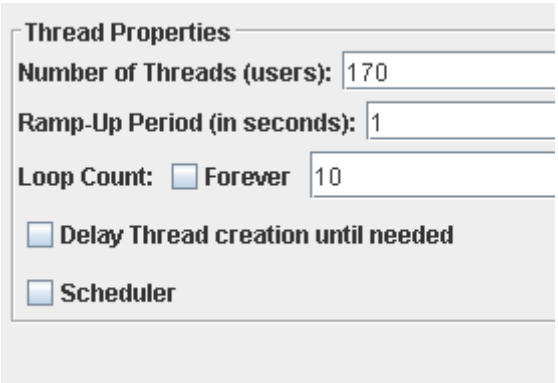


As we can check with next reports, there aren't any errors and response time is acceptable. So, this is the maximum workload supported by the system.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throu...	KB/sec
/security/login.do	1600	406	155	1105	11	7754	0.00%	5.5/sec	19.8
/j_spring_security_check	1600	804	553	1804	35	8415	0.00%	5.5/sec	21.8
/	3200	417	184	1082	12	8069	0.00%	10.3/s...	36.8
/actor/display.do	1600	457	224	1144	18	5789	0.00%	5.5/sec	28.2
/socialProfile/administrator,auditor,customer,nutritionist,trainer/list.do	8000	469	224	1208	17	9576	0.00%	26.2/s...	116.7
/socialProfile/administrator,auditor,customer,nutritionist,trainer/create.do	1600	526	240	1287	21	7968	0.00%	5.5/sec	26.7
/socialProfile/administrator,auditor,customer,nutritionist,trainer/edit.do	8000	594	314	1502	12	10585	0.00%	26.5/s...	193.2
/socialProfile/administrator,auditor,customer,nutritionist,trainer/display.do	1600	888	562	2114	22	7925	0.00%	5.5/sec	21.5
/j_spring_security_logout	1600	387	165	1056	14	5774	0.00%	5.5/sec	19.6
TOTAL	28800	534	262	1383	11	10585	0.00%	92.4/s...	467.2

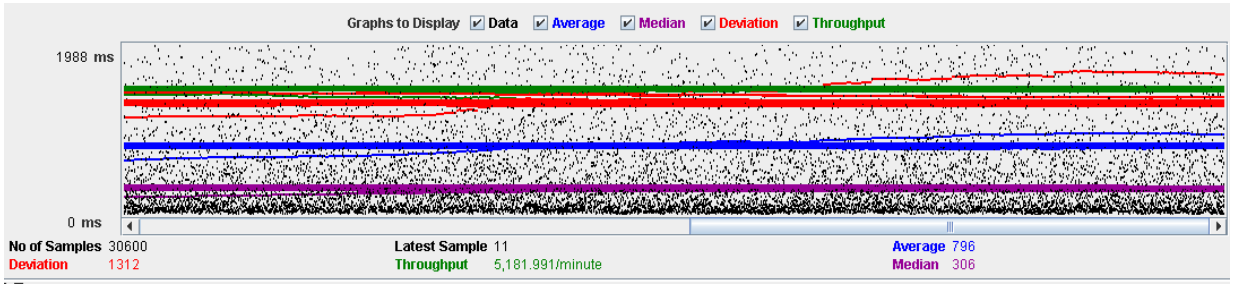


Overload test case: 170 concurrent users, 10 of loop count and 1 as ramp-up period.



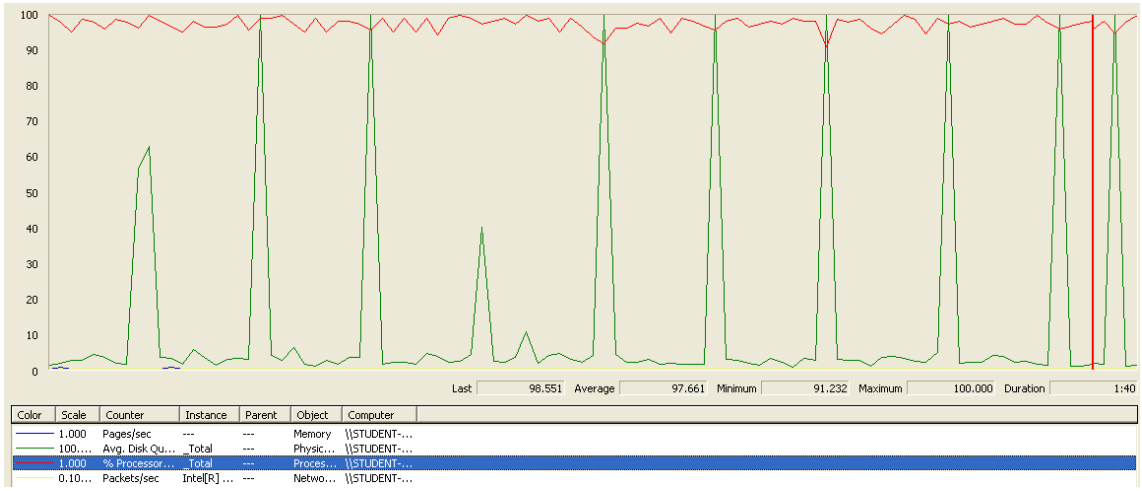
The request with highest response time is "display.do". We could to try refactoring the code to improve the performance.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throu...	KB/sec
/security/login.do	1700	488	164	1336	9	7319	0.00%	5.3/sec	19.3
/j_spring_security_check	1700	1050	678	2478	27	7960	0.00%	5.3/sec	21.1
/	3400	525	227	1353	10	8382	0.00%	10.1/s...	35.9
/actor/display.do	1700	565	244	1463	21	8701	0.00%	5.3/sec	27.4
/socialProfile/administrator,auditor,customer,nutritionist,trainer/list.do	8500	595	257	1520	17	11853	0.00%	25.6/s...	113.9
/socialProfile/administrator,auditor,customer,nutritionist,trainer/create.do	1700	587	232	1548	19	10654	0.00%	5.3/sec	25.9
/socialProfile/administrator,auditor,customer,nutritionist,trainer/edit.do	8500	756	358	1995	14	12288	0.00%	25.8/s...	188.4
/socialProfile/administrator,auditor,customer,nutritionist,trainer/display.do	1700	1128	642	2883	20	9304	0.00%	5.3/sec	21.0
/j_spring_security_logout	1700	508	206	1320	13	8614	0.00%	5.4/sec	19.1
TOTAL	30600	674	299	1771	9	12288	0.00%	90.4/s...	457.2



It’s obvious that there is a bottleneck in the CPU. It is saturated (CPU is working at 90%-100% all the time).

Besides, there are some peaks in hard disk. It occurs when the system retrieves social profiles list from the database. The hard disk is not powerful enough.



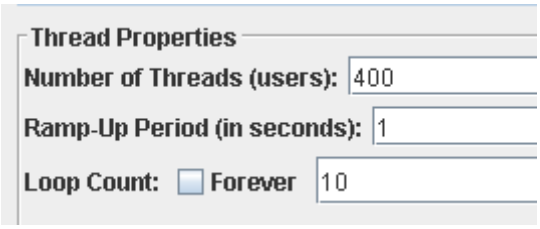
Conclusion: The maximum number of concurrent users supported by the system is 160. There are bottlenecks in CPU and hard disk. Replacing them by a more powerful components might boost the system. It’s not necessary to update the other components. Besides, we can refactor code when the system must to display a social profile.

Req. 11.1 - An actor who is not authenticated as an administrator must be able to: Create user accounts for new administrators.

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

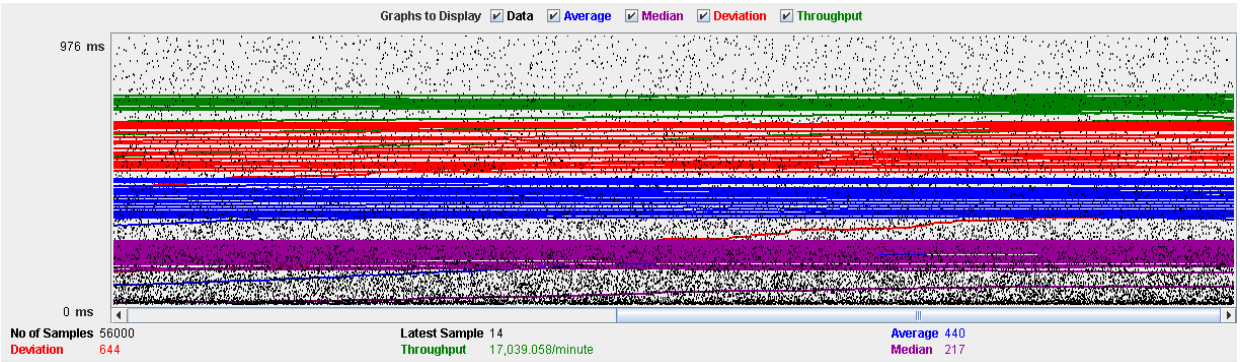
Test case description: The user logs in the system as an administrator, then, he/she clicks on Register administrator option in the main menu and fills the form. When the user completes the form, press Save and then logs out. Finally, the user logs in with the new user account.

Maximum workload test case: 400 concurrent users, 10 of loop count and 1 of ramp-up period.

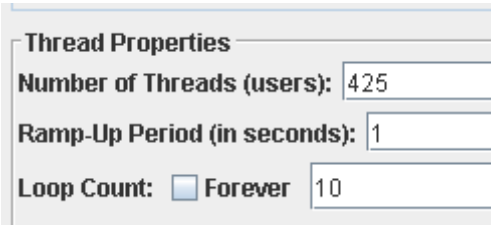


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	20000	378	184	955	3	9153	0.00%	101.4/sec	373.
/security/login.do	8000	375	168	945	2	11556	0.00%	41.5/sec	148.
/j_spring_security_c...	8000	743	474	1730	6	9026	0.00%	41.5/sec	180.
/actor/administrator/...	8000	420	210	1025	5	9133	0.00%	42.8/sec	347.
/welcome/index.do	4000	409	201	1043	4	6980	0.00%	21.6/sec	89.
/j_spring_security_l...	8000	394	192	1013	4	9899	0.00%	42.3/sec	151.
TOTAL	56000	440	217	1113	2	11556	0.00%	284.0/sec	1253.



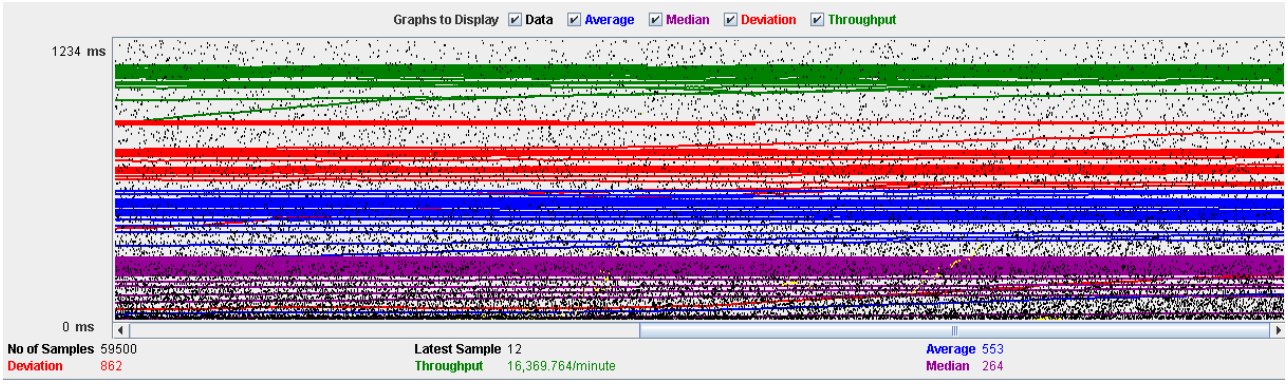
Overload test case: 425 concurrent users, 10 of loop count and 1 as ramp-up period.



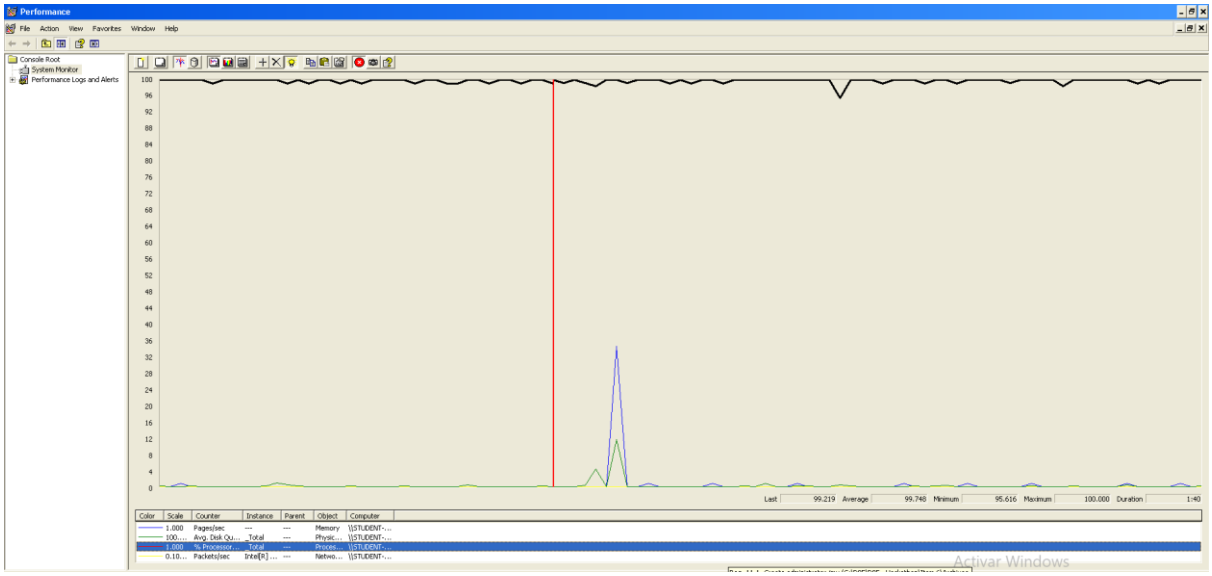
Even though the average time per request is still acceptable, it begins to produce some errors as we can see in the following picture. The errors are always the same: I/O exception (java.net.SocketException) caught when processing request: Connection reset by peer: socket write error.

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
/	21250	469	220	1169	0	18761	0.52%	97.4/sec	358.0
/security/login.do	8500	471	205	1135	0	18023	0.38%	39.9/sec	142.1
/j_spring_security_c...	8500	943	591	2147	0	12794	0.79%	39.8/sec	172.7
/actor/administrator/...	8500	514	252	1285	0	13692	0.41%	40.9/sec	329.6
/welcome/index.do	4250	532	254	1320	0	11817	0.40%	20.6/sec	84.7
/j_spring_security_l...	8500	501	240	1235	0	14235	0.61%	40.4/sec	144.5
TOTAL	59500	553	264	1373	0	18761	0.53%	272.8/sec	1199.9



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.



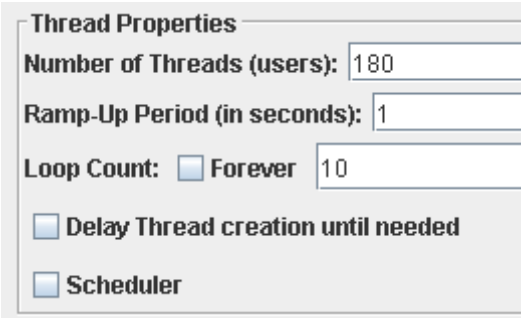
Conclusion: The maximum number of concurrent users supported by the system is 400. There is a bottleneck in CPU. Replacing it by a more powerful components might boost the system. It's not necessary to update the other components.

Req. 11.2 - An actor who is authenticated as administrator must be able to manage the catalogue of categories, which includes listing, creating, updating and deleting them. Note that categories evolve independently from working-outs, which means that they can be created or modified independently from whether they are referenced from a working-out or not, it can only be deleted if there is no reference.

- 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 + 1TB GB HDD
- Network card: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter.

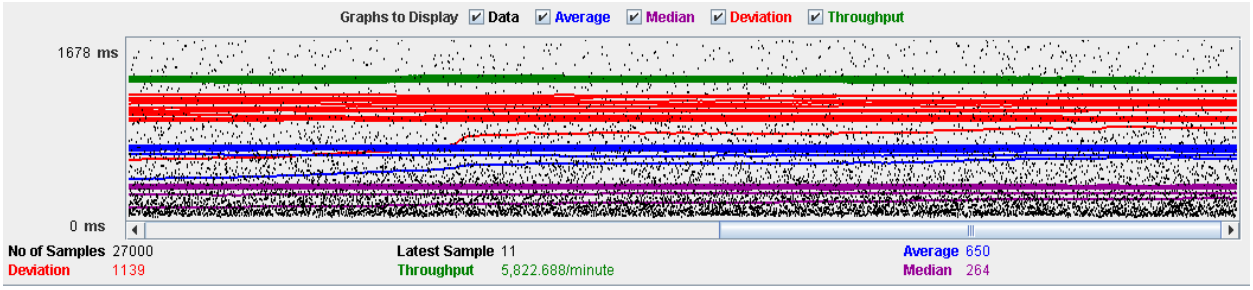
Test case description: First, the user login as an administrator ("admin1"). Then, he or she clicks "Categories list" link. Later, the user clicks "Create a new category" link. Immediately, the user fills the form and press save button. Below, he or she edits the newly category. Now, a category is displayed. Straightaway, the user deletes the edited category. Eventually, the user logs out.

Maximum workload test case: 180 concurrent users, 10 of loop count and 1 of ramp-up period.

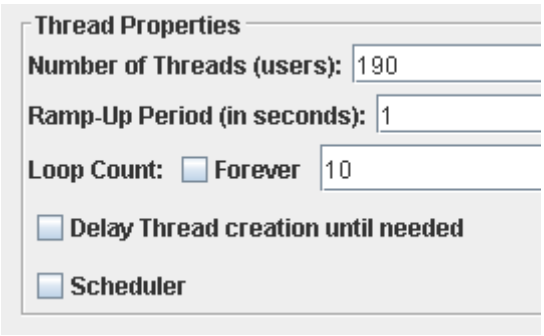


According to the performance test, this is the maximum workload that can be supported by the system without errors/failures or insufficient performance.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1800	457	178	1081	10	10670	0.00%	6.7/sec	24.4
/j_spring_securit...	1800	1043	617	2422	20	15471	0.00%	6.7/sec	29.1
/	3600	528	203	1266	10	13400	0.00%	13.0/sec	48.6
/category/admini...	7200	572	213	1395	10	15656	0.00%	26.2/sec	147.6
/category/admini...	1800	534	199	1286	13	14861	0.00%	6.8/sec	32.8
/category/admini...	9000	779	337	1902	15	16908	0.00%	33.2/sec	158.2
/j_spring_securit...	1800	482	197	1190	13	9146	0.00%	6.9/sec	24.5
TOTAL	27000	650	264	1597	10	16908	0.00%	97.0/sec	454.0

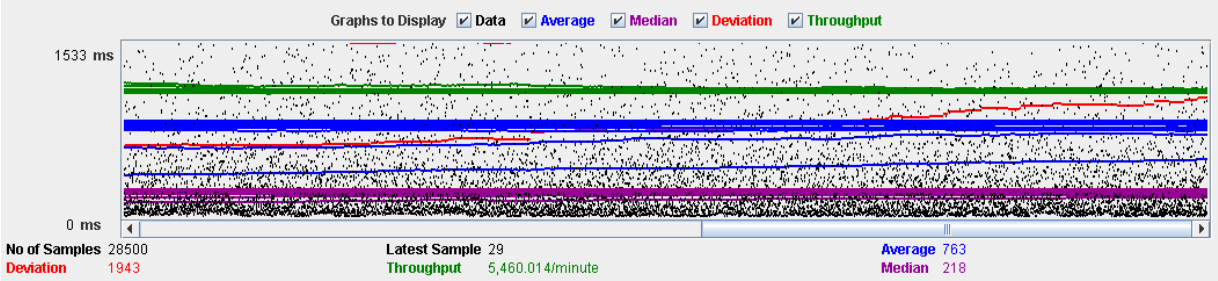


Overload test case: 190 concurrent users, 10 of loop count and 1 as ramp-up period.



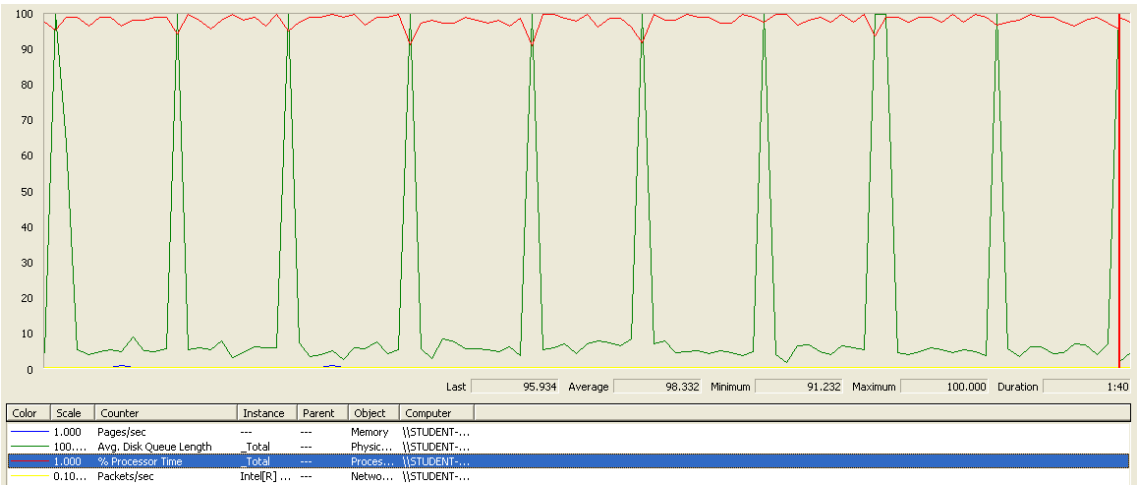
The guilty that system cannot support a higher workload is the unit known as “j_spring_security_check”. This is an internal unit of Spring. That’s means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1900	561	136	1075	9	20507	0.00%	6.4/sec	23.1
/j_spring_securit...	1900	1238	461	2654	19	37201	0.00%	6.4/sec	27.6
/	3800	647	170	1264	9	22707	0.00%	12.2/sec	45.5
/category/admini...	7600	636	169	1213	10	26921	0.00%	24.6/sec	138.6
/category/admini...	1900	681	187	1167	12	33068	0.00%	6.4/sec	30.9
/category/admini...	9500	895	277	1821	13	32547	0.00%	31.1/sec	148.1
/j_spring_securit...	1900	651	182	1313	13	17248	0.00%	6.4/sec	22.7
TOTAL	28500	763	218	1494	9	37201	0.00%	91.0/sec	425.8



It’s obvious that there is a bottleneck in the CPU. It is saturated (CPU is working at 90%-100% all the time).

Besides, there are some peaks in hard disk. It occurs when the system retrieves social profiles list from the database. There is a bottleneck in hard disk too.



Conclusion: The maximum number of concurrent users supported by the system is 180. There are bottlenecks in CPU and hard disk. Replacing them by a more powerful components might boost the system. It’s not necessary to update the other components.

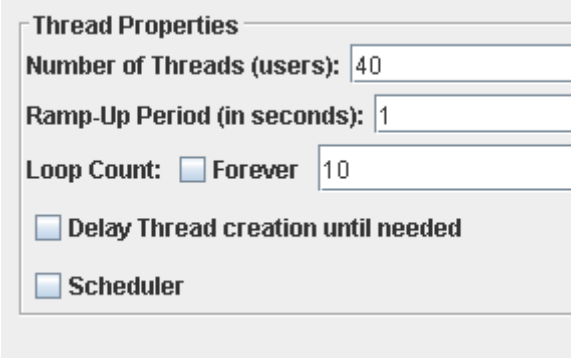
Req. 11.3 - An actor who is authenticated as administrator must be able to broadcast a message to all the actors of the system.

- 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 + 1TB GB HDD

- Network card: Ethernet: Realtek PCIe GBE Family Controller | Wi-Fi: Qualcomm Atheros QCA9377 Wireless Network Adapter.

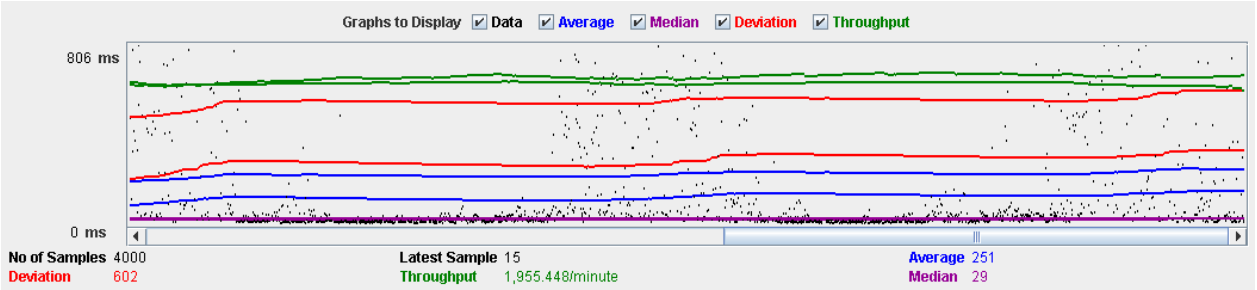
Test case description: First, the user login as an administrator ("admin1"). Then, he or she clicks "Boxes list" link. Later, the user clicks "Send a broadcast message" link. Immediately, the user fills the form and press send button. Eventually, the user logs out.

Maximum workload test case: 40 concurrent users, 10 of loop count and 1 of ramp-up period.

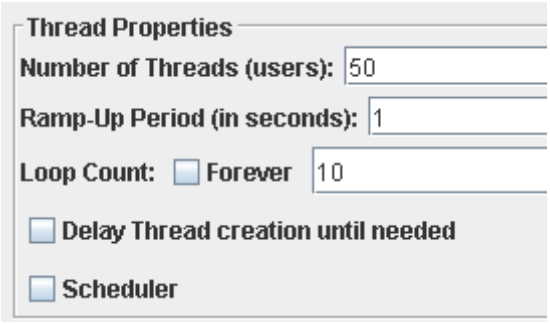


According to the performance test, this is the maximum workload that can be supported by the system: there aren't any errors and response time per request is acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	1200	98	18	122	9	3749	0.00%	9.8/sec	35.4
/security/login.do	400	45	12	25	8	2935	0.00%	3.7/sec	13.2
/j_spring_security_check	400	65	27	49	17	2145	0.00%	3.7/sec	16.1
/box/administrator,auditor,customer,nutritionist,trainer/list.do	800	259	36	834	14	4450	0.00%	6.9/sec	38.7
/message/administrator/broadcast.do	800	673	309	1849	24	5373	0.00%	7.0/sec	39.0
/j_spring_security_logout	400	247	33	675	13	4523	0.00%	3.6/sec	12.6
TOTAL	4000	251	29	825	8	5373	0.00%	32.6/sec	145.2

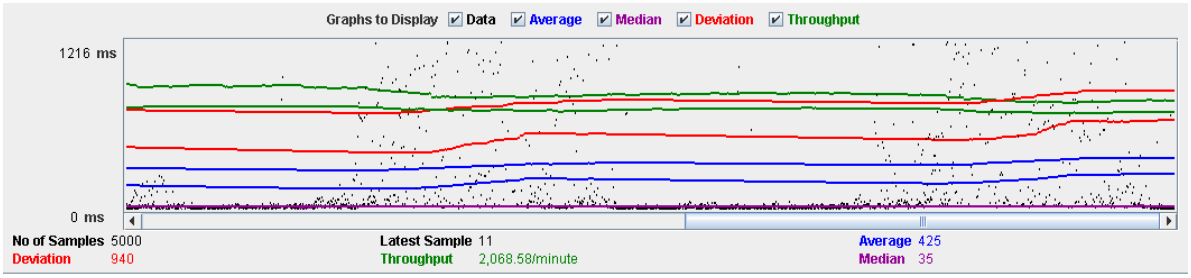


Overload test case: 50 concurrent users, 10 of loop count and 1 as ramp-up period.



There is a lot of difference in terms of response time between the request “send a broadcast message” and the other requests. When a message (notification or not) is sent, the system must check if this message contains any spam word. This check is very inefficient. So, if we can get refactoring the code, the system would support a higher workload.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	1500	169	22	405	8	5791	0.00%	10.3/sec	37.5
/security/login.do	500	77	13	84	7	2220	0.00%	3.6/sec	12.9
/j_spring_security_check	500	135	28	82	17	6081	0.00%	3.6/sec	15.7
/box/administrator,auditor,customer,nutritionist,trainer/list.do	1000	428	43	1378	15	6591	0.00%	7.1/sec	39.6
/message/administrator/broadcast.do	1000	1107	569	3088	26	7520	0.00%	7.1/sec	39.5
/j_spring_security_logout	500	464	47	1533	13	5191	0.00%	3.6/sec	12.8
TOTAL	5000	425	35	1425	7	7520	0.00%	34.5/sec	153.6



The CPU work at 80% sometimes, but we can’t consider the CPU as a bottleneck.

Although, we can consider the hard disk as a bottleneck because it has peaks at 100%. It occurs when the system retrieves boxes list from the database.

Conclusion: The maximum number of concurrent users supported by the system is 40. There is a bottleneck with hard disk. Replacing it by a more powerful component might boost the system. Besides, we can increase the maximum workload if we refactoring the code when system checks if a broadcast message contains spam words.

Req. 24 – Trainers can register their curriculum. Every curriculum has a ticker, a personal record, some education records, some professional records, some endorser records, and some miscellaneous records.

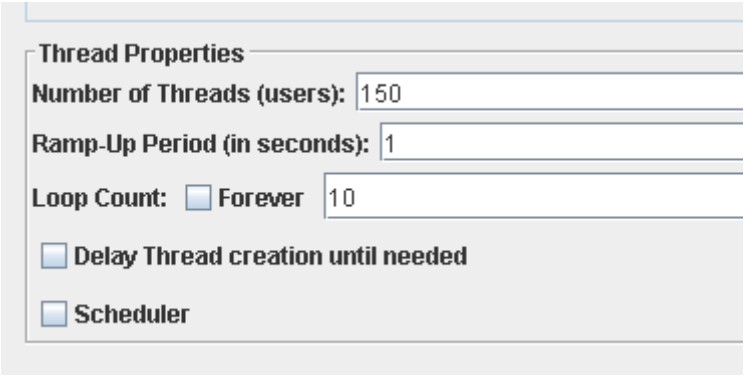
Technical details of the computer on which the test has been executed:

- 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:

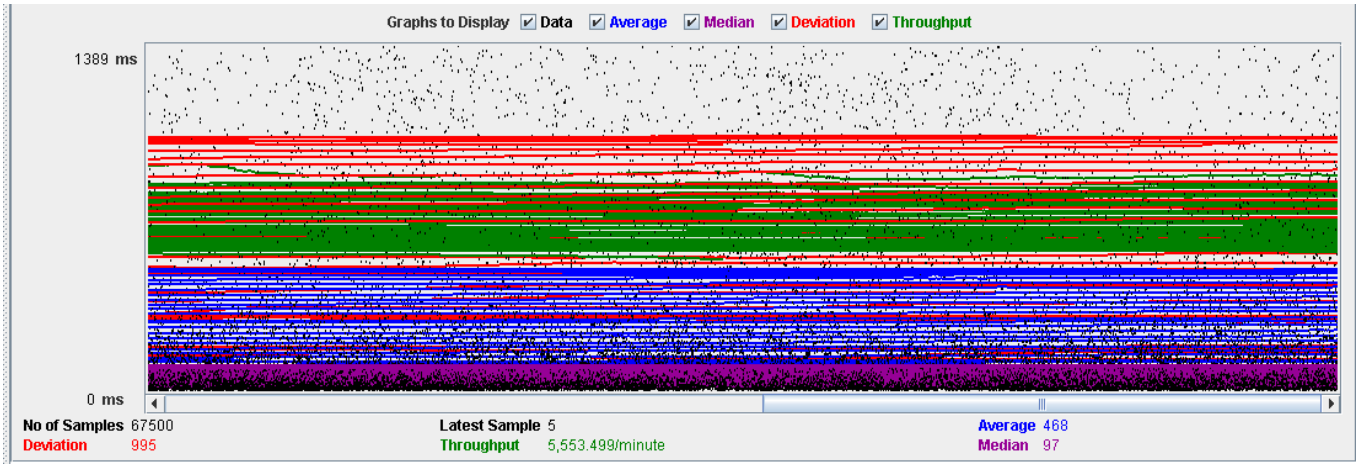
- Log in as trainer
- Display his curriculum
- Edit the personal record of the curriculum
- Create an endorser record in the curriculum
- Edit the new endorser record
- Delete the endorser record
- Create an education record
- Edit the new education record
- Delete the education record
- Create a professional record in the curriculum
- Edit the new professional record
- Delete the professional record
- Create a miscellaneous record in the curriculum
- Edit the new miscellaneous record
- Delete the miscellaneous record
- Log out

Maximum workload test case. 150 concurrent users and 10 of loop count

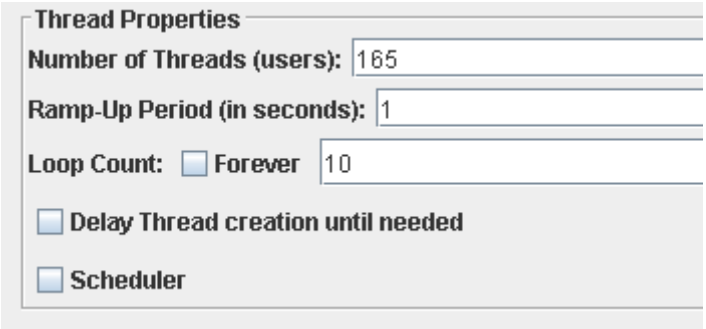


This is the maximum workload of the test case without any crash or excessive delay. As we can see in the picture below, we don't have any errors and the average time per request is acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1500	353	46	1071	4	9021	0.00%	2.2/sec	7.9
/j_spring_security_check	1500	490	121	1189	9	11120	0.00%	2.2/sec	8.7
/	3000	274	47	529	4	13221	0.00%	4.1/sec	14.6
/curriculum/trainer/display.do	21000	334	90	767	8	17112	0.00%	28.9/sec	432.7
/personalRecord/trainer/edit.do	3000	393	57	1294	7	11791	0.00%	4.4/sec	24.8
/endorserRecord/trainer/create.do	1500	395	44	1311	5	10579	0.00%	2.2/sec	11.6
/endorserRecord/trainer/edit.do	7500	539	90	1749	5	18705	0.00%	10.8/sec	53.3
/educationRecord/trainer/create.do	1500	451	62	1495	5	8981	0.00%	2.2/sec	12.2
/educationRecord/trainer/edit.do	7500	635	149	1991	6	20764	0.00%	10.8/sec	61.3
/professionalRecord/trainer/create.do	1500	466	65	1704	5	7542	0.00%	2.2/sec	12.1
/professionalRecord/trainer/edit.do	7500	639	160	2027	6	15518	0.00%	10.8/sec	61.5
/miscellaneousRecord/trainer/create.do	1500	415	59	1435	4	12961	0.00%	2.2/sec	10.7
/miscellaneousRecord/trainer/edit.do	7500	597	157	1892	5	18518	0.00%	10.8/sec	58.6
/j_spring_security_logout	1500	421	68	1361	6	11459	0.00%	2.2/sec	7.8
TOTAL	67500	468	97	1431	4	20764	0.00%	92.6/sec	758.7



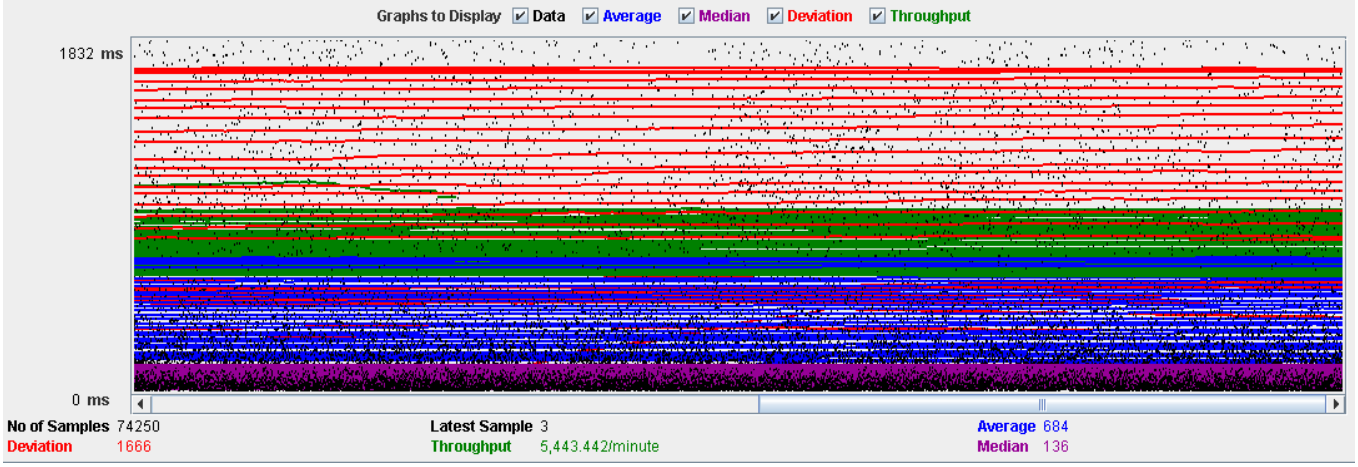
Overload test case: 165 concurrent users and 10 of loop count:



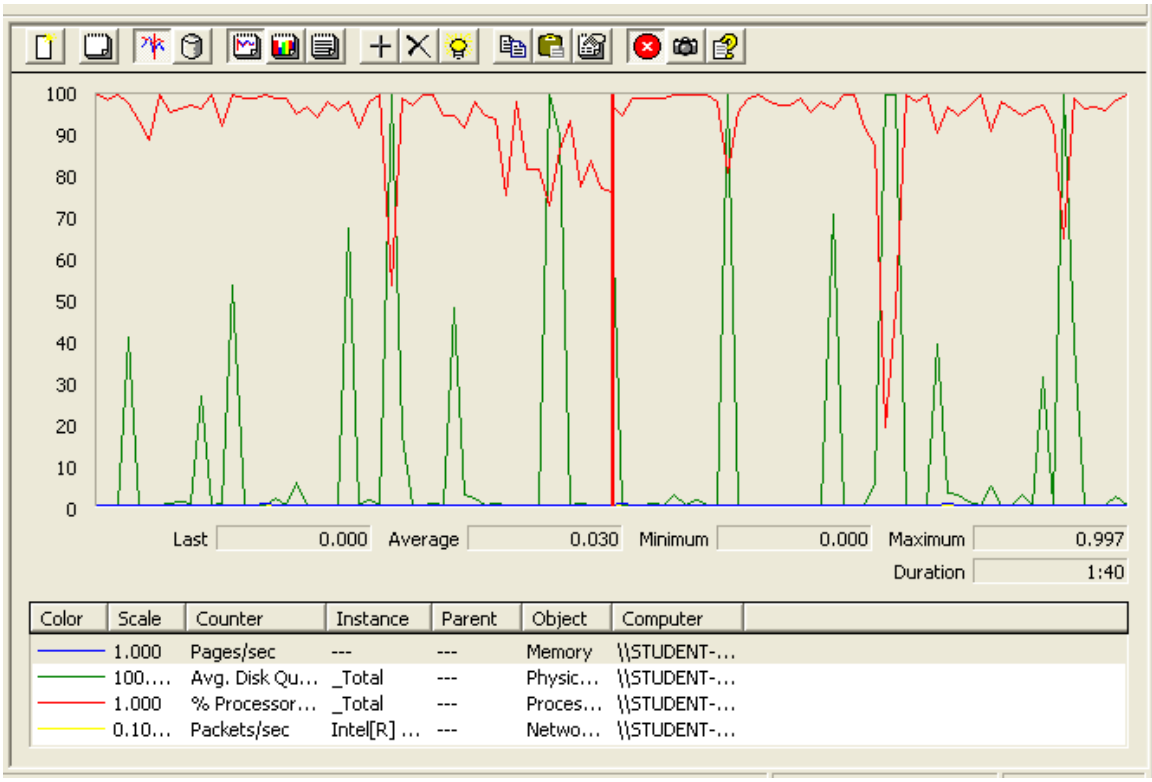
We don’t have any errors, but the system works with excessive delay.

As we can see in the table below, the request with the highest average time is /miscellaneousRecord/trainer/edit.do. This URI correspond to the edition of a miscellaneous record of the curriculum.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1650	513	72	1610	3	19946	0.00%	2.1/sec	7.7
/j_spring_security_check	1650	812	181	1833	9	21046	0.00%	2.1/sec	8.5
/	3300	417	71	716	3	19813	0.00%	4.0/sec	14.4
/curriculum/trainer/display.do	23100	523	118	1155	7	35345	0.00%	28.3/sec	424.2
/personalRecord/trainer/edit.do	3300	535	87	1524	7	25841	0.00%	4.2/sec	24.1
/endorserRecord/trainer/create.do	1650	615	96	1993	4	21717	0.00%	2.1/sec	11.3
/endorserRecord/trainer/edit.do	8250	756	137	2109	5	30461	0.00%	10.6/sec	52.0
/educationRecord/trainer/create.do	1650	604	87	1802	5	15266	0.00%	2.1/sec	11.8
/educationRecord/trainer/edit.do	8250	887	219	2424	6	33130	0.00%	10.6/sec	59.6
/professionalRecord/trainer/create.do	1650	705	96	2031	6	20805	0.00%	2.1/sec	11.8
/professionalRecord/trainer/edit.do	8250	884	205	2407	6	29957	0.00%	10.6/sec	59.5
/miscellaneousRecord/trainer/create.do	1650	636	99	1921	5	18073	0.00%	2.1/sec	10.5
/miscellaneousRecord/trainer/edit.do	8250	890	207	2526	5	30076	0.00%	10.6/sec	56.8
/j_spring_security_logout	1650	589	85	1626	7	32283	0.00%	2.1/sec	7.6
TOTAL	74250	684	136	1886	3	35345	0.00%	90.7/sec	743.0



As we can see in the graphic below, there are some spikes of CPU and hard drive. Both are relevant since both kind of spikes reach 100% of capability sometimes, especially the CPU which reach 100% very frequently.



Conclusion: The maximum number of concurrent users supported by this test case is 150. In order to improve the performance, the first thing we have to do is analyzing the code of the URIs with the most avarage time per request, which are:

- /educationRecord/trainer/edit.do
- /professional/trainer/edit.do
- /miscellaneousRecord/trainer/edit.do

These requests are very similar, so there may be some line of codes which are not very efficient.

After that, we can try to improve the performance by assigning more CPU's resources to our system, but we don't expect this to improve too much the performance of the system, since the CPU capability was reaching 100% sometimes and not in a continuous way.

Req. 31 – Auditors can audit the curricula of the trainers. For every audit, the system must store a record with the moment when the audit was carried out, a title, a description, and some optional attachments.

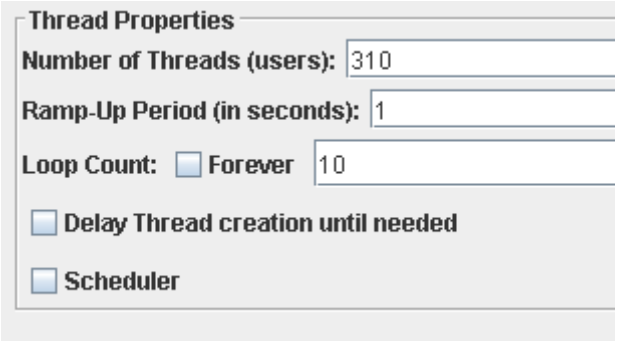
Technical details of the computer on which the test has been executed:

- 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:

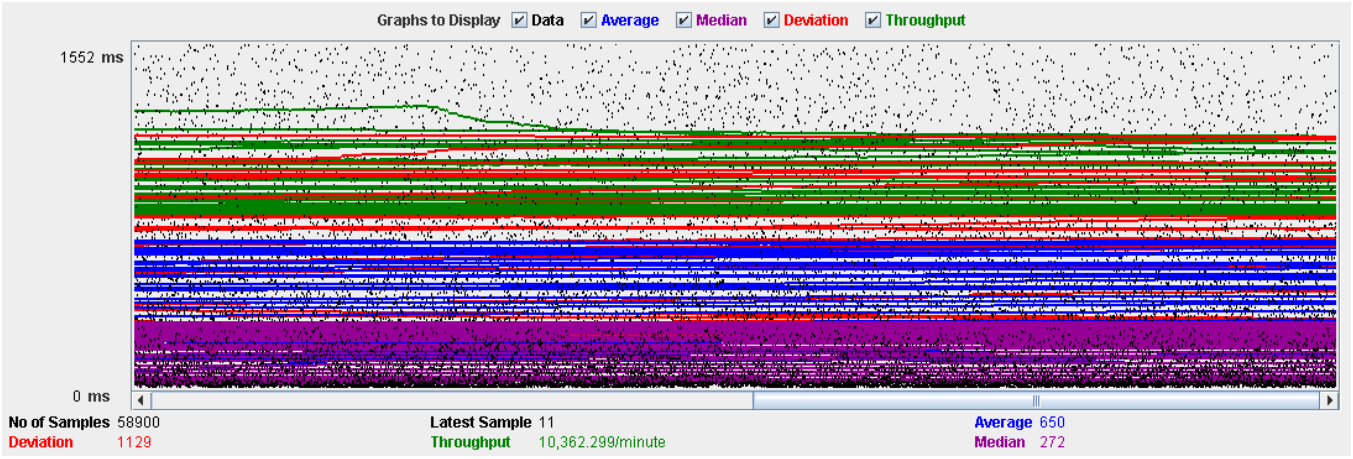
- Log in as auditor
- List the trainer of the system
- Display the profile of one of the trainers
- Display the curriculum of the trainer
- Create an audit about the curriculum displayed
- Once the audit is saved, the system redirects to the audit list
- Edit the new audit and save it
- Edit the updated audit and delete it
- Log out

Maximum workload test case. 310 concurrent users and 10 of loop count

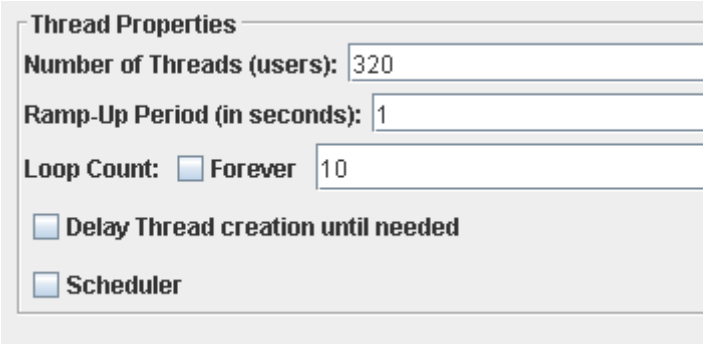


This is the maximum workload of the test case without any crash or excessive delay. As we can see in the picture below, we don't have any errors and the average time per request is acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	3100	511	214	1273	3	16600	0.00%	9.5/sec	34.8
/j_spring_security_check	3100	948	566	2226	8	21674	0.00%	9.5/sec	36.3
/	6200	454	188	1061	4	21339	0.00%	18.2/sec	63.1
/trainer/auditor/list.do	3100	593	232	1458	7	15892	0.00%	9.5/sec	46.8
/actor/display.do	3100	557	196	1401	6	16097	0.00%	9.5/sec	45.0
/curriculum/displayFromTrainer.do	3100	560	220	1474	8	14891	0.00%	9.5/sec	64.1
/audit/auditor/create.do	3100	605	234	1687	5	12228	0.00%	9.5/sec	44.5
/audit/auditor/edit.do	15500	778	357	1996	4	29288	0.00%	46.6/sec	184.3
/audit/auditor/display.do	6200	914	512	2198	7	16264	0.00%	18.9/sec	70.7
/audit/auditor/list.do	9300	476	175	1226	6	15757	0.00%	28.1/sec	136.2
/j_spring_security_logout	3100	522	211	1316	6	18087	0.00%	9.6/sec	34.4
TOTAL	58900	650	272	1658	3	29288	0.00%	172.7/sec	734.0



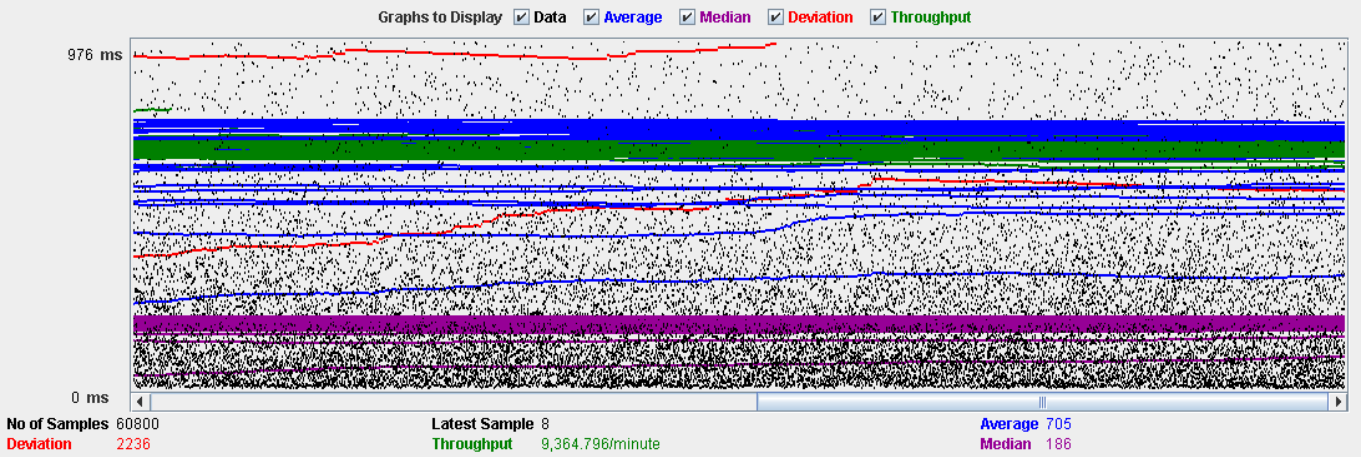
Overload test case: 320 concurrent users and 10 of loop count:



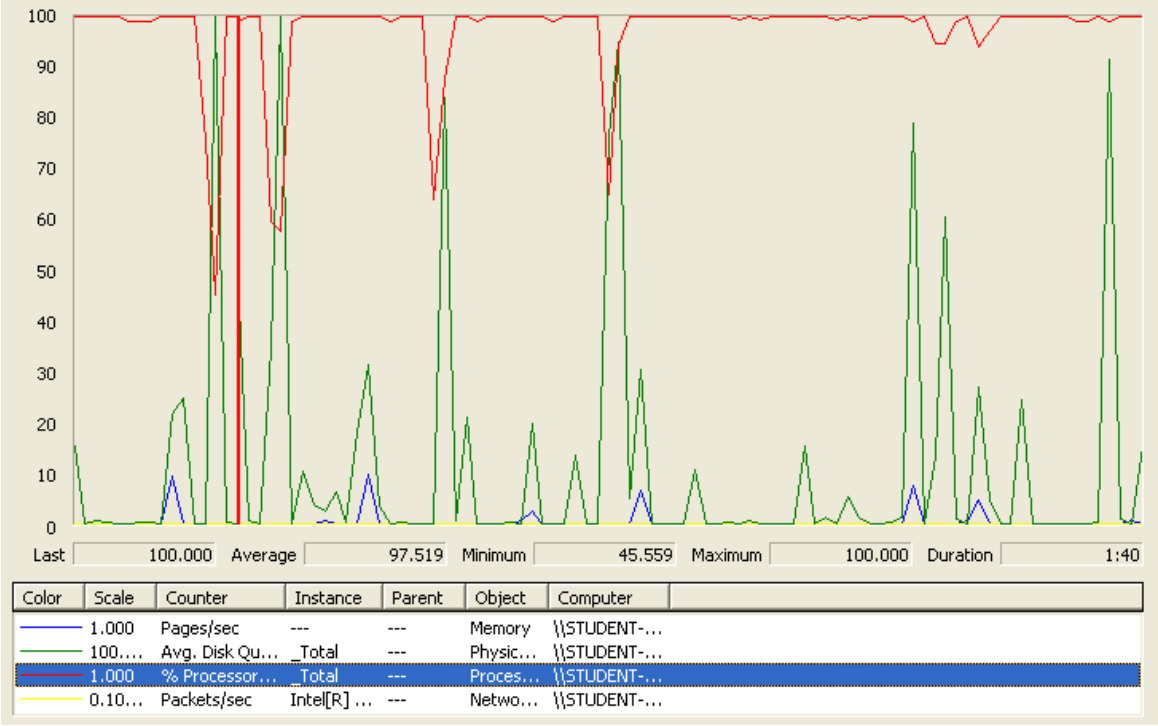
Even though the average time per request is still acceptable, it begins to produce some errors as we can see in the following picture. The errors are always the same: `I/O exception (java.net.SocketException) caught when processing request: Connection reset by peer: socket write error`

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
/security/login.do	3200	560	131	809	2	32382	0.28%	8.6/sec	31.4
/j_spring_security_check	3200	1124	323	1808	1	39300	1.06%	8.6/sec	32.7
/	6400	461	121	623	1	30547	0.62%	16.4/sec	56.8
/trainer/auditor/list.do	3200	537	147	749	1	33186	0.06%	8.6/sec	42.3
/actor/display.do	3200	549	147	834	8	42490	0.09%	8.6/sec	40.7
/curriculum/displayFromTrainer.do	3200	583	174	829	10	22473	0.09%	8.6/sec	57.9
/audit/auditor/create.do	3200	548	152	816	2	41946	0.09%	8.7/sec	40.3
/audit/auditor/edit.do	16000	843	241	1470	5	40727	0.08%	42.2/sec	166.7
/audit/auditor/display.do	6400	1099	321	2058	1	39766	0.12%	17.1/sec	64.1
/audit/auditor/list.do	9600	528	134	712	3	38407	0.09%	25.4/sec	123.0
/j_spring_security_logout	3200	580	135	797	5	31485	0.06%	8.7/sec	31.3
TOTAL	60800	705	186	1077	1	42490	0.21%	156.1/sec	662.4



In the graph below we can clearly see that there is a bottleneck in the CPU which is at 100% of capability most of the time. There are some spikes in the hard drive too, but they are not relevant since they rarely reach the 100% of capability.



Conclusion: The maximum number of concurrent users supported by this test case is 320 and we could improve it by assigning more CPU's resources to our system. However, is possible that the exception thrown during the execution of the test (`java.net.SocketException`) doesn't dissappear, since is an exception related with tomcat.

So, if increasing CPU's resources doesn't improve the performance, updating tomcat component or assigning more resources to tomcat component could be a good idea.

Req. 32 – Customers have a finder in which they can specify some filters: a single key word that must appear somewhere in the ticker, description or address of the sessions; a category to which the working-out must belong; a range of prices; or a range of dates. The finder stores the working-outs that pass the filters for one hour by default.

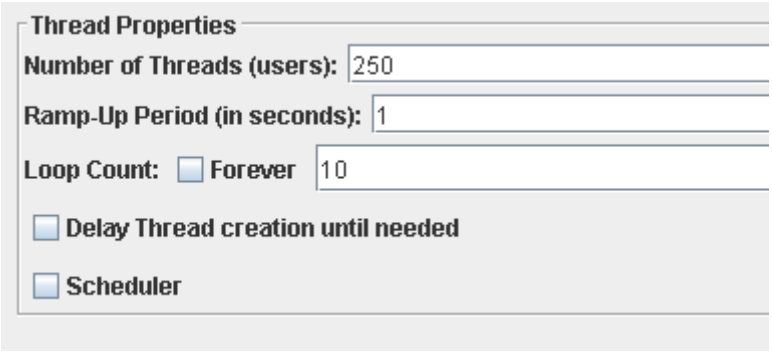
Technical details of the computer on which the test has been executed:

- 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:

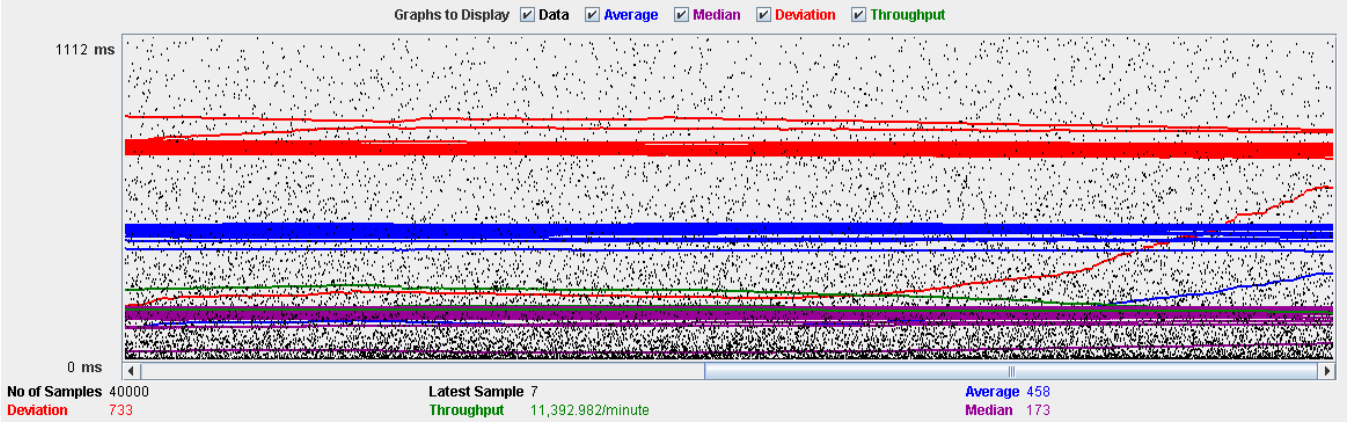
- Log in as customer
- Display his finder
- Clear the finder clicking on “Clear” link
- Edit the finder, set the key word with “14” and the start date with “01/01/2020” and save the changes
- Edit the finder and click on the “Clear” button
- Log out

Maximum workload test case. 250 concurrent users and 10 of loop count

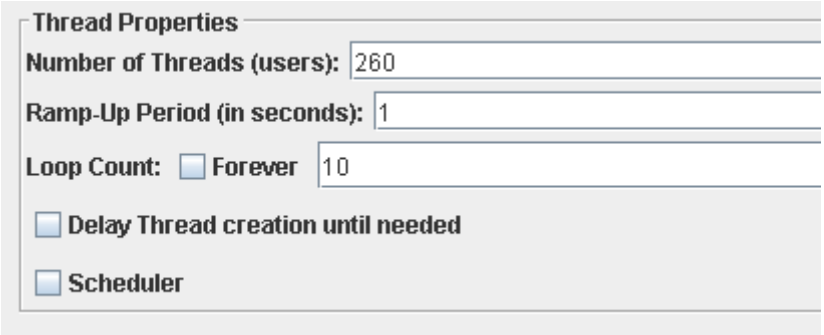


This is the maximum workload of the test case without any crash or excessive delay. As we can see in the picture below, we don’t have any errors and the average time per request is acceptable.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/images/arrow_down.png	2500	5	3	6	0	624	0.00%	12.6/sec	5.6
/security/login.do	2500	377	147	1027	3	5760	0.00%	12.6/sec	45.7
/j_spring_security_check	2500	665	402	1572	9	6152	0.00%	12.6/sec	57.1
/	5000	325	139	830	4	6810	0.00%	23.9/sec	91.8
/finder/customer/display.do	10000	446	183	1161	8	8160	0.00%	48.4/sec	440.0
/images/arrow_off.png	2500	7	3	5	0	639	0.00%	12.6/sec	5.6
/finder/customer/clear.do	5000	807	437	2060	9	7569	0.00%	24.4/sec	275.9
/finder/customer/edit.do	7500	605	273	1646	7	6964	0.00%	37.2/sec	275.0
/j_spring_security_logout	2500	419	176	1162	6	7010	0.00%	12.7/sec	45.5
TOTAL	40000	458	173	1274	0	8160	0.00%	189.9/sec	1204.4



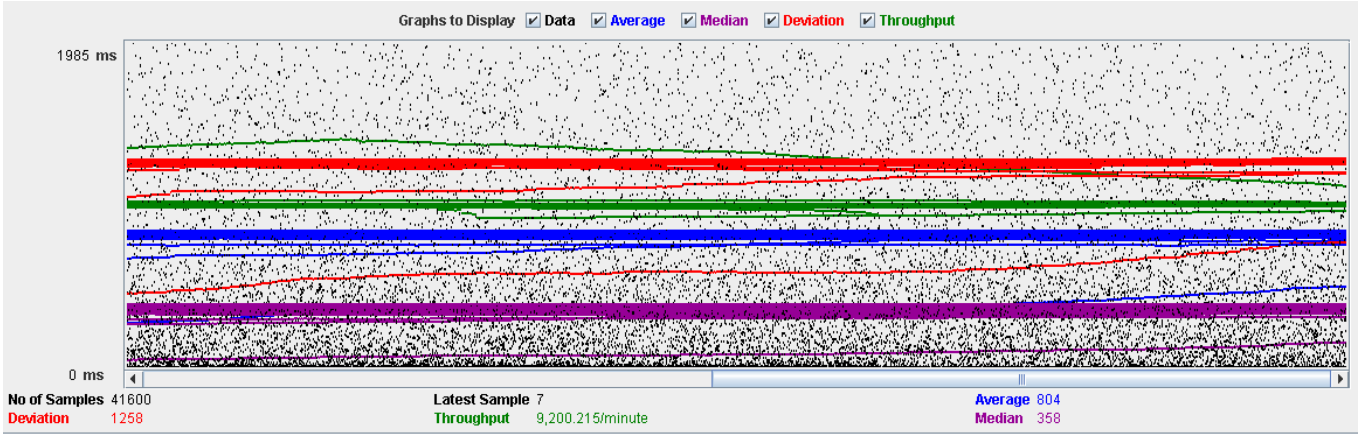
Overload test case: 260 concurrent users and 10 of loop count:



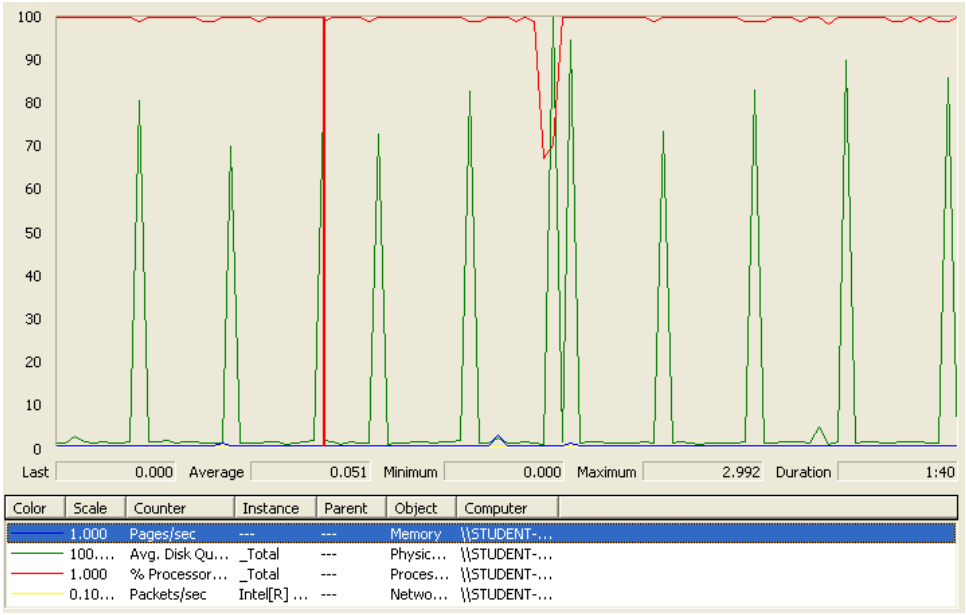
We don't have any errors, but the system works with excessive delay.

As we can see in the table below, the request with the highest average time are /j_spring_security_check (this URI correspond to an internal implementation of spring, so we cannot improve the performance by refactoring the code) and /finder/customer/clear.do (this URI is used to clear all the fields of a finder).

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/images/arrow_down.png	2600	5	4	7	0	556	0.00%	10.0/sec	4.5
/security/login.do	2600	726	341	1886	4	10883	0.00%	10.0/sec	36.4
/j_spring_security_check	2600	1489	1014	3364	8	15942	0.00%	10.0/sec	45.6
/	5200	701	334	1757	4	13868	0.00%	19.3/sec	74.1
/finder/customer/display.do	10400	751	370	1859	8	19511	0.00%	38.9/sec	345.0
/images/arrow_off.png	2600	4	4	7	0	103	0.00%	10.0/sec	4.5
/finder/customer/clear.do	5200	1336	818	3167	10	14231	0.00%	19.6/sec	217.2
/finder/customer/edit.do	7800	935	471	2288	7	18124	0.00%	29.8/sec	221.4
/j_spring_security_logout	2600	763	365	1833	6	12469	0.00%	10.2/sec	36.4
TOTAL	41600	804	358	2092	0	19511	0.00%	153.3/sec	961.3



In the graph below we can clearly see that there is a bottleneck in the CPU which is at 100% of capability most of the time. There are some spikes in the hard drive too, but they are not relevant since they rarely reach the 100% of capability.



Conclusion: The maximum number of concurrent users supported by this test case is 250. In order to improve the performance, we only way is assigning more CPU resources to our system because:

- CPU is working at 100% most of the time.
- We cannot improve the performance of the URI /j_spring_security_check since this one corresponds to an internal implementation of spring.
- We cannot improve the performance of the URI /finder/customer/clear.do since the implementation is really simple: get the finder of the customer principal and set all the values to null or empty string.

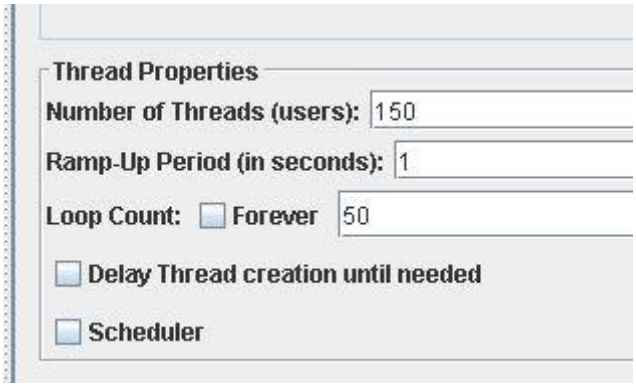
Additional Performance Tests

Req. 9.1 - An actor who is authenticated as a trainer must be able to list and display working-outs.

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

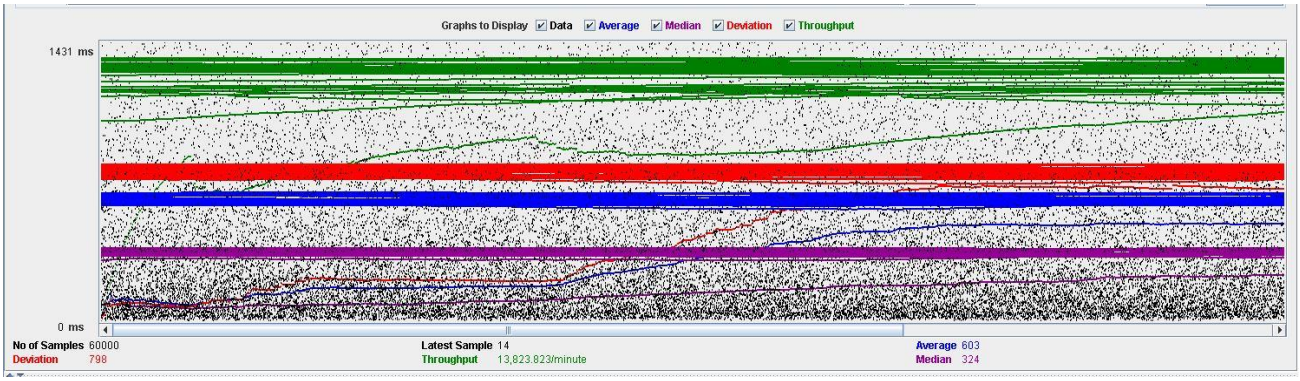
Test case description: The user login as a trainer, go to working-outs list and display a working-out. The user logout to the system.

Maximum workload test case: 150 concurrent users, 50 of loop count and 1 of ramp-up period.

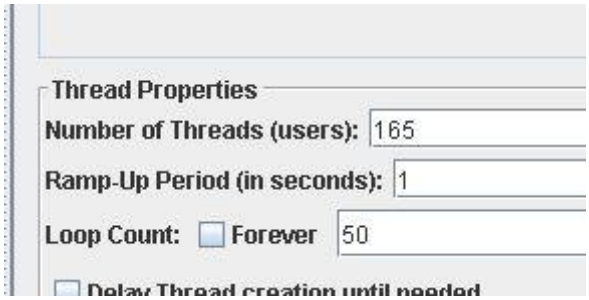


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	22500	533	282	1331	5	14196	0.00%	86.4/sec	301.5
/security/login.do	7500	529	274	1346	4	10237	0.00%	28.8/sec	102.5
/j_spring_security_check	7500	1084	740	2427	10	13038	0.00%	28.8/sec	114.9
/workingOut/trainer/list.do	7500	519	273	1290	8	14937	0.00%	28.8/sec	135.6
/workingOut/customer,tra...	7500	546	306	1324	14	9376	0.00%	28.9/sec	173.5
/j_spring_security_logout	7500	545	285	1339	10	10857	0.00%	28.9/sec	102.5
TOTAL	60000	603	324	1491	4	14937	0.00%	230.4/sec	929.4

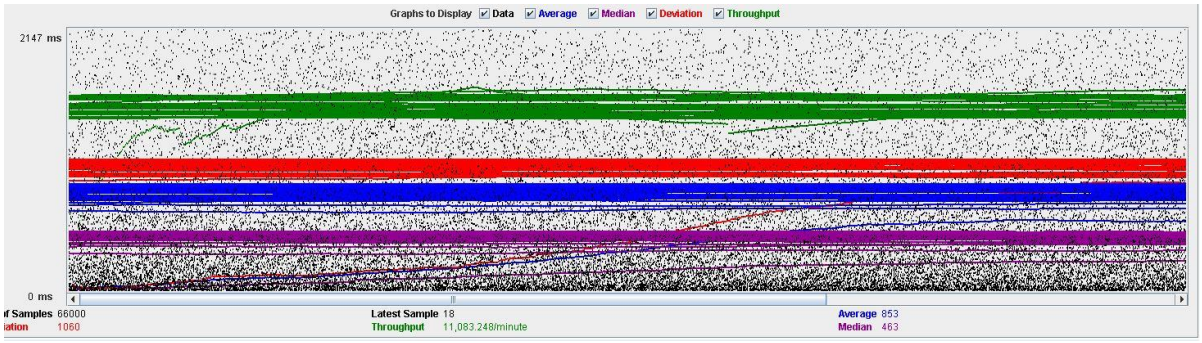


Overload test case: 165 concurrent users, 50 of loop count and 1 as ramp-up period.



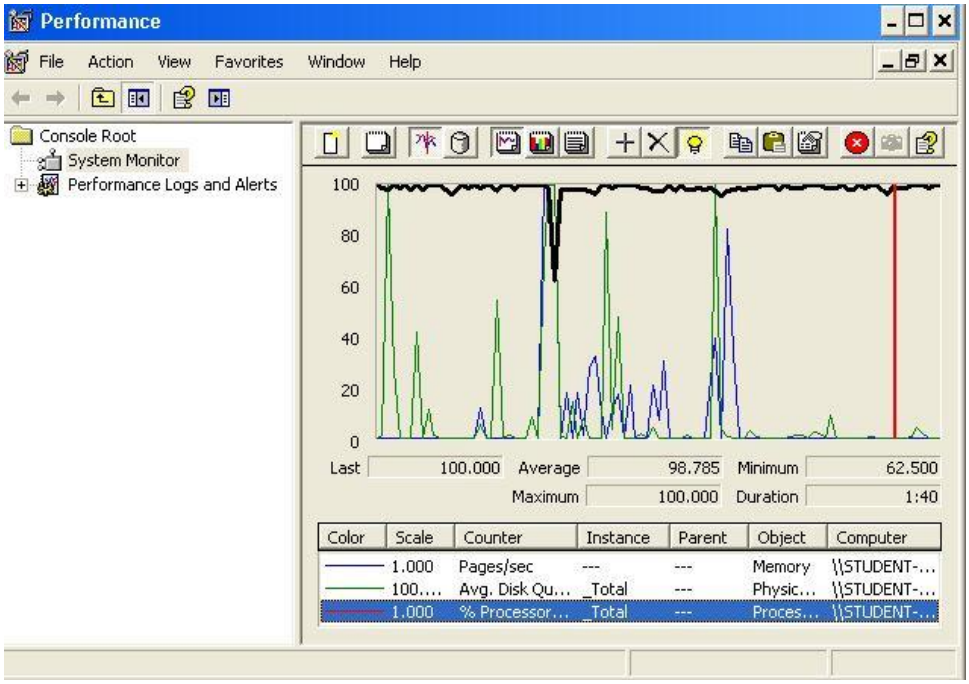
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	24750	745	399	1957	3	9400	0.00%	69.3/sec	242.8
/securityllogin.do	8250	762	401	1982	3	10467	0.00%	23.1/sec	82.5
/j_spring_security_check	8250	1541	1129	3426	8	15235	0.00%	23.1/sec	92.8
/workingOut/trainer/list.do	8250	759	409	1995	4	8397	0.00%	23.1/sec	109.0
/workingOut/customer,tra...	8250	773	422	1961	10	9089	0.00%	23.1/sec	139.3
/j_spring_security_logout	8250	750	401	1968	4	11096	0.00%	23.1/sec	82.8
TOTAL	66000	853	463	2210	3	15235	0.00%	184.7/sec	748.9



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.

In this analysis we can also see that hard disk make some peak. We think that is when the system goes to memory to get the list.



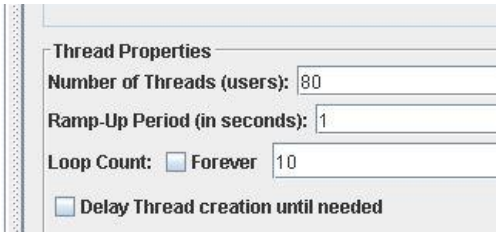
Conclusion: The maximum number of concurrent users supported by the system is 150 because the current CPU is a bottleneck.

Req. 9.1 - An actor who is authenticated as a trainer must be able to create and delete working-outs.

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

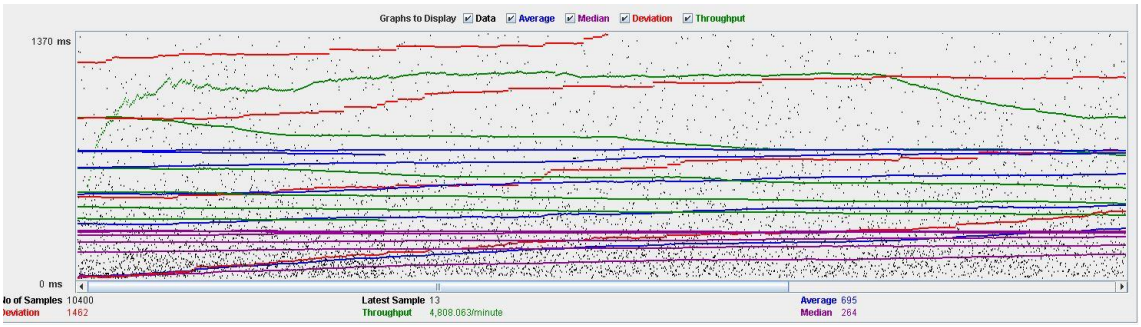
Test case description: The user login as a trainer, go to working-outs create and delete a working-out. The user creates new working-out, later click on edit link and delete the working-out created previously. The user logout to the system.

Maximum workload test case: 80 concurrent users, 10 of loop count and 1 of ramp-up period.

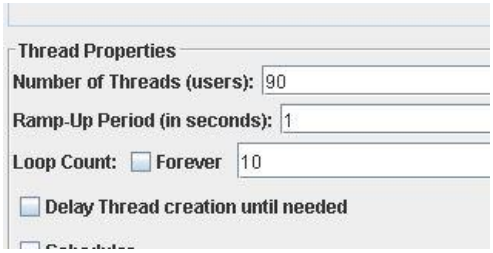


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	2400	525	182	1207	3	16999	0.00%	18.5/sec	64.6
/security/login.do	800	546	178	1239	6	8572	0.00%	6.2/sec	22.1
/j_spring_security_check	800	1093	506	2332	11	21153	0.00%	6.2/sec	24.7
/workingOut/trainer/list.do	2400	640	241	1308	7	17205	0.00%	18.5/sec	126.5
/workingOut/trainer/creat...	800	501	185	1094	9	10529	0.00%	6.2/sec	41.1
/workingOut/trainer/edit.do	2400	953	429	2144	10	30745	0.00%	18.5/sec	113.1
/j_spring_security_logout	800	539	183	1139	18	16049	0.00%	6.2/sec	22.0
TOTAL	10400	695	264	1549	3	30745	0.00%	80.1/sec	412.9

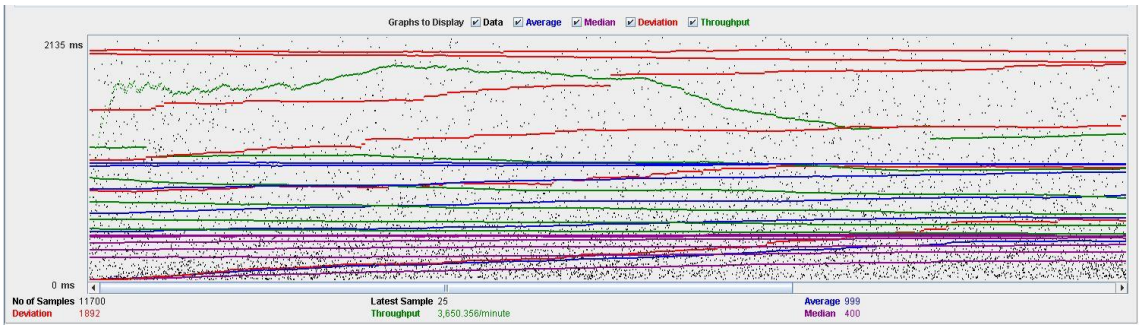


Overload test case: 90 concurrent users, 10 of loop count and 1 as ramp-up period.

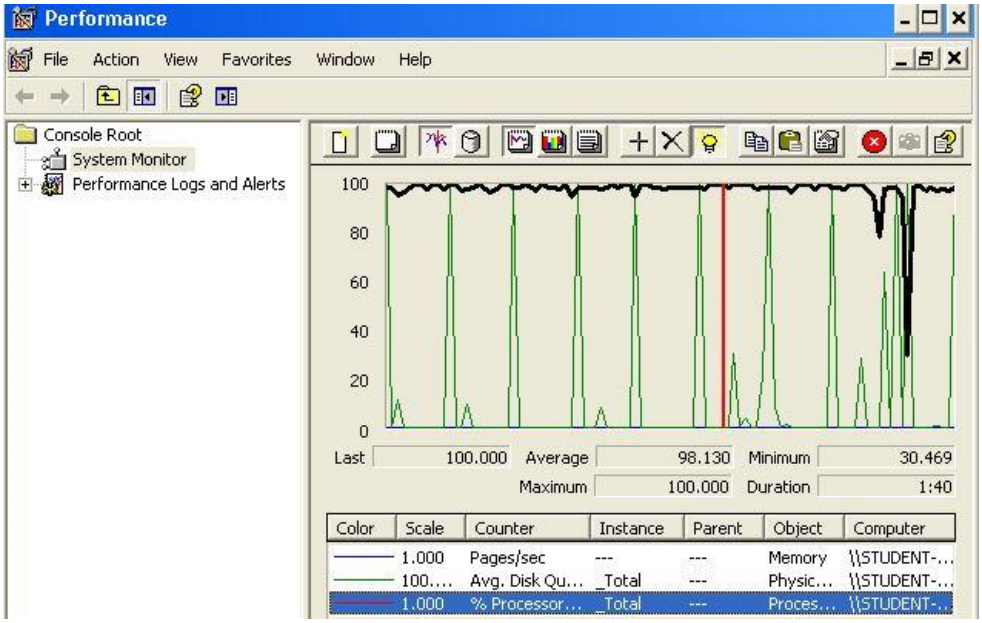


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

	2700	734	291	1772	3	22816	0.00%	14.0/sec	49.0
security/login.do	900	743	270	1545	4	16988	0.00%	4.7/sec	16.7
j_spring_security_check	900	1497	759	3575	11	23654	0.00%	4.7/sec	18.7
workingOut/trainer/list.do	2700	883	366	1968	12	27858	0.00%	14.0/sec	97.2
workingOut/trainer/creat...	900	768	301	1807	12	17678	0.00%	4.7/sec	31.1
workingOut/trainer/edit.do	2700	1456	669	3482	13	48209	0.00%	14.1/sec	86.2
j_spring_security_logout	900	761	305	1799	18	18722	0.00%	4.7/sec	16.7
TOTAL	11700	999	400	2292	3	48209	0.00%	60.8/sec	315.1



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.



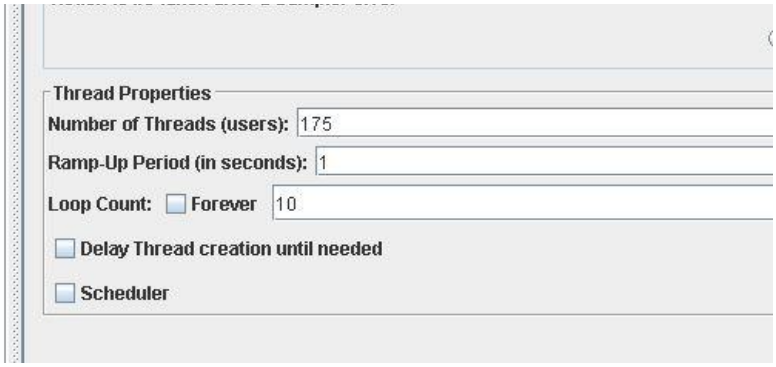
Conclusion: The maximum number of concurrent users supported by the system is 80 because the current CPU is a bottleneck.

Req. 9.1 - An actor who is authenticated as a trainer must be able to edit and make final their working-outs.

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

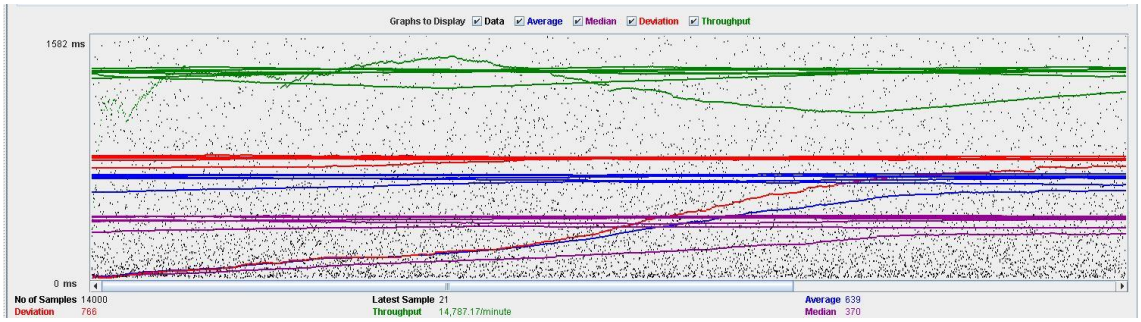
Test case description: The user login as a trainer, go to working-outs edit a parameter and save form. Later make final this working-out. The user logout to the system.

Maximum workload test case: 175 concurrent users, 10 of loop count and 1 of ramp-up period.

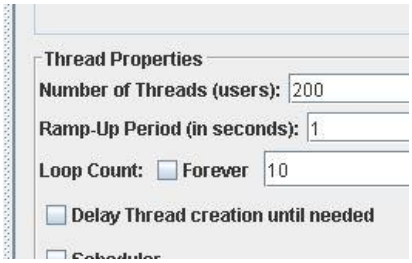


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/j_spring_security_check	1750	1011	705	2275	9	7457	0.00%	30.9/sec	134.4
/	3500	509	278	1331	4	5307	0.00%	61.6/sec	230.7
/customisation/administr...	3500	512	275	1294	5	5465	0.00%	61.7/sec	457.9
/customisation/administr...	3500	786	513	1891	6	6223	0.00%	61.8/sec	504.5
/j_spring_security_logout	1750	489	260	1240	5	4247	0.00%	30.9/sec	110.7
TOTAL	14000	639	370	1603	4	7457	0.00%	246.5/sec	1435.4

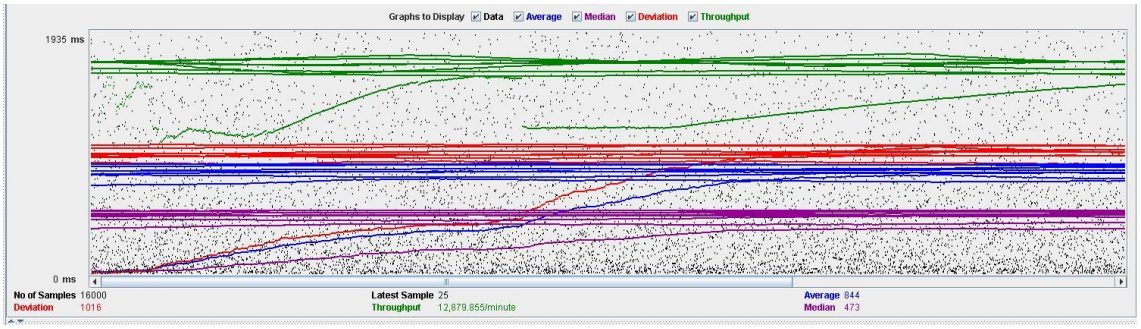


Overload test case: 200 concurrent users, 10 of loop count and 1 as ramp-up period.

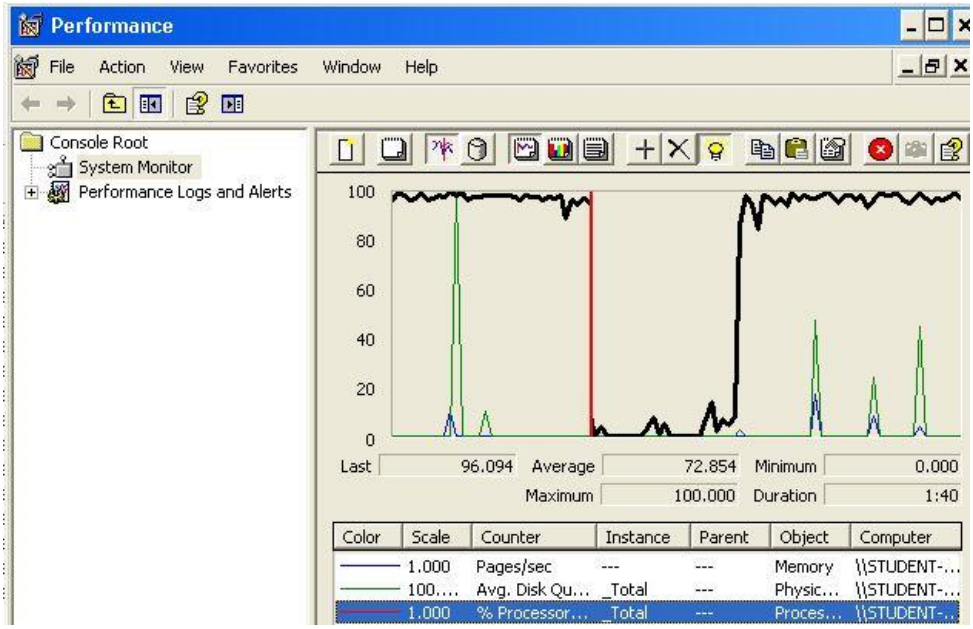


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/j_spring_security_check	2000	1354	985	3040	11	9227	0.00%	26.9/sec	117.0
/	4000	649	342	1670	4	9985	0.00%	53.7/sec	201.1
/customisation/administr...	4000	662	348	1711	5	8244	0.00%	53.8/sec	398.9
/customisation/administr...	4000	1054	669	2622	5	7948	0.00%	53.8/sec	439.4
/j_spring_security_logout	2000	670	357	1760	5	8545	0.00%	26.9/sec	96.4
TOTAL	16000	844	473	2195	4	9985	0.00%	214.7/sec	1250.6



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.



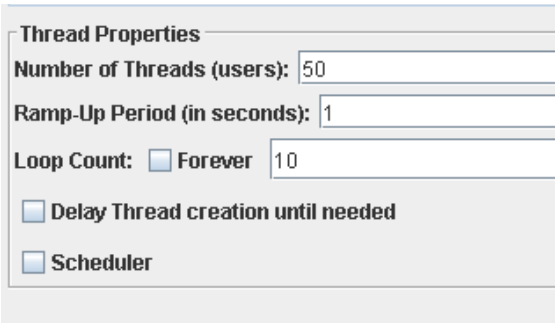
Conclusion: The maximum number of concurrent users supported by the system is 80 because the current CPU is a bottleneck.

Ρεθ. 9.2 – An actor who is authenticated as a trainer must be able to manage the applications for his or her working-outs, which includes listing and updating them. A trainer may update an application to change its status from pending to either accepted or rejected.

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

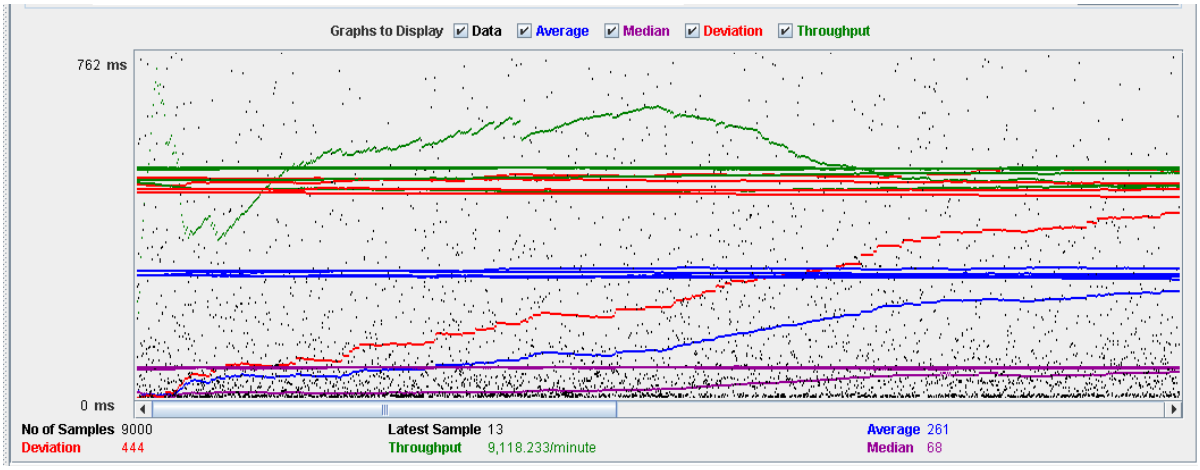
Test case description: The user trainer initiates a session, then lists the working-outs, shows a working-outs, accesses the list of applications and updates any of them. Finally show a application and close session.

Maximum workload test case: 50 concurrent users, 10 of loop count and 1 of ramp-up period.

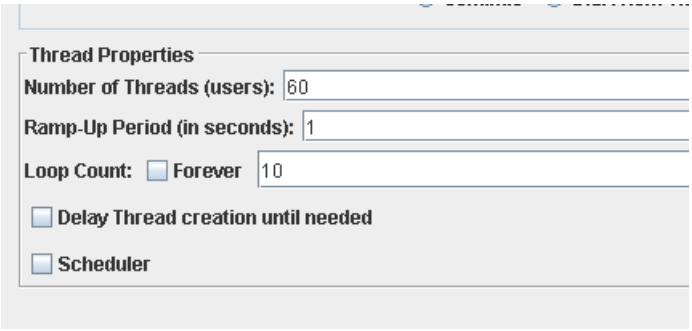


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	500	302	144	798	10	4240	0.00%	8.5/sec	30.4
/scripts/jquery-ui...	500	47	42	65	13	231	0.00%	8.5/sec	3884.4
/scripts/md5-mi...	500	7	6	11	1	232	0.00%	8.6/sec	46.1
/styles/jmenu.css	500	7	6	11	2	328	0.00%	8.5/sec	16.1
/scripts/jmenu.js	500	9	6	12	2	506	0.00%	8.5/sec	90.3
/styles/common....	500	7	6	11	2	211	0.00%	8.5/sec	4.9
/scripts/jquery.js	500	29	27	42	7	91	0.00%	8.5/sec	2301.1
/styles/displayta...	500	8	6	11	1	437	0.00%	8.5/sec	25.5
/j_spring_securit...	500	658	444	1477	22	3516	0.00%	8.5/sec	34.0
/	1000	356	179	970	11	3314	0.00%	16.9/sec	60.3
/workingOut/train...	500	330	156	880	13	4077	0.00%	8.6/sec	40.2
/workingOut/cust...	500	441	241	1000	33	3788	0.00%	8.6/sec	55.4
/application/train...	1000	392	205	1007	22	5070	0.00%	17.2/sec	82.0
/application/train...	500	677	473	1523	25	3887	0.00%	8.7/sec	34.1
/application/cust...	500	341	151	969	15	3371	0.00%	8.8/sec	36.4
/j_spring_securit...	500	340	167	867	6	2795	0.00%	8.8/sec	31.3
TOTAL	9000	261	68	804	1	5070	0.00%	152.0/sec	6687.2

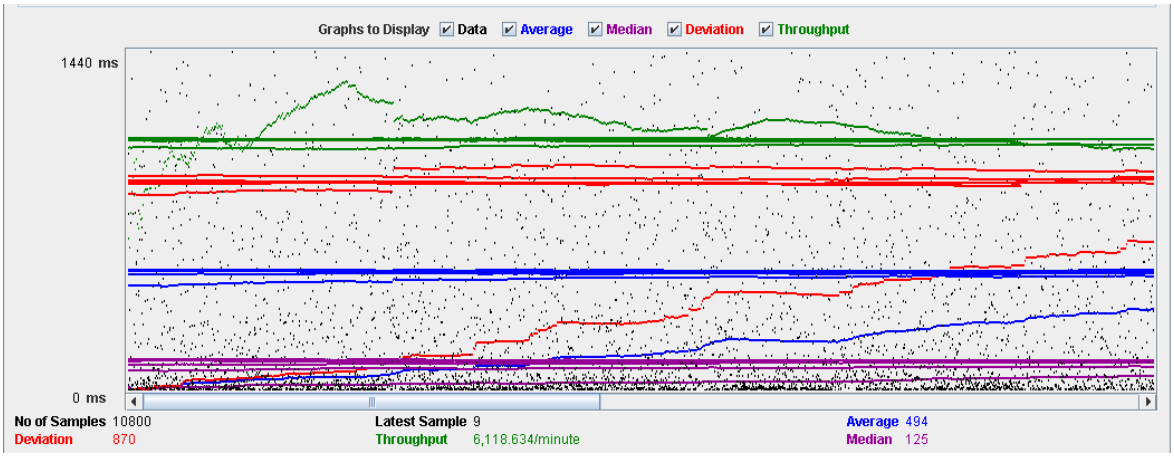


Overload test case: 60 concurrent users, 10 of loop count and 1 as ramp-up period.



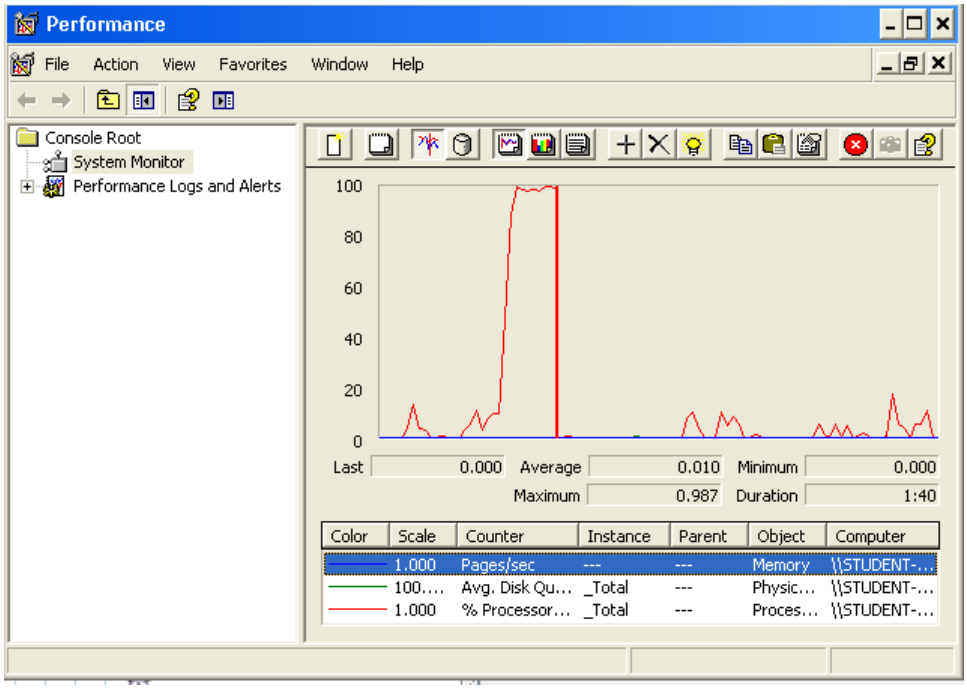
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	600	642	357	1573	10	9941	0.00%	5.7/sec	20.3
/scripts/jquery-ui...	600	69	54	105	7	761	0.00%	5.7/sec	2594.6
/scripts/md5-mi...	600	11	9	22	2	162	0.00%	5.7/sec	30.8
/styles/jmenu.css	600	11	9	22	1	140	0.00%	5.7/sec	10.7
/scripts/jmenu.js	600	12	9	23	2	195	0.00%	5.7/sec	60.3
/styles/common....	600	12	9	22	1	566	0.00%	5.7/sec	3.3
/scripts/jquery.js	600	40	35	66	5	340	0.00%	5.7/sec	1537.5
/styles/displayta...	600	12	9	20	2	668	0.00%	5.7/sec	17.0
/j_spring_securit...	600	1331	903	2896	20	19503	0.00%	5.7/sec	22.7
/	1200	661	353	1588	9	10445	0.00%	11.3/sec	40.4
/workingOut/train...	600	637	356	1425	15	8010	0.00%	5.7/sec	26.8
/workingOut/cust...	600	667	439	1450	31	5975	0.00%	5.7/sec	36.9
/application/train...	1200	749	414	1777	21	8989	0.00%	11.4/sec	54.3
/application/train...	600	1283	956	2756	26	10104	0.00%	5.7/sec	22.4
/application/cust...	600	687	399	1665	14	6667	0.00%	5.7/sec	23.6
/j_spring_securit...	600	660	348	1664	15	6014	0.00%	5.7/sec	20.3
TOTAL	10800	494	125	1424	1	19503	0.00%	102.0/sec	4487.3



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.

In this analysis we can also see that hard disk make some peak. We think that is when the system goes to memory to get the list.



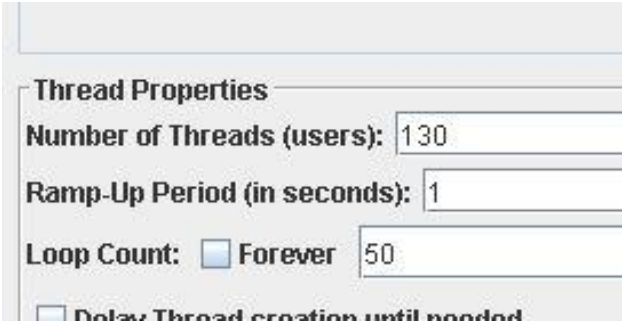
Conclusion: The maximum number of concurrent users supported by the system is 50 because the current CPU is a bottleneck.

Req. 10.1 - An actor who is authenticated as a customer must be able browser the catalogue of working-outs and navigate to the profile of the corresponding trainer.

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

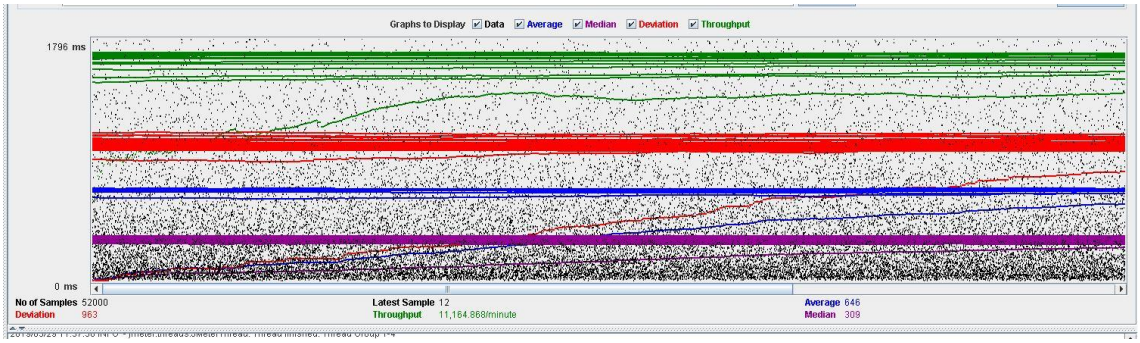
Test case description: The user login as a customer, go to working-outs list and display a working-out. When he/she is in display view click on trainer name link. Finally, the user logout.

Maximum workload test case: 130 concurrent users, 50 of loop count and 1 of ramp-up period.

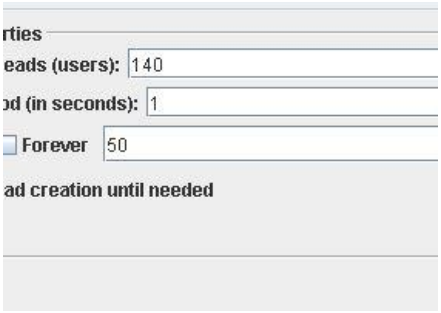


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

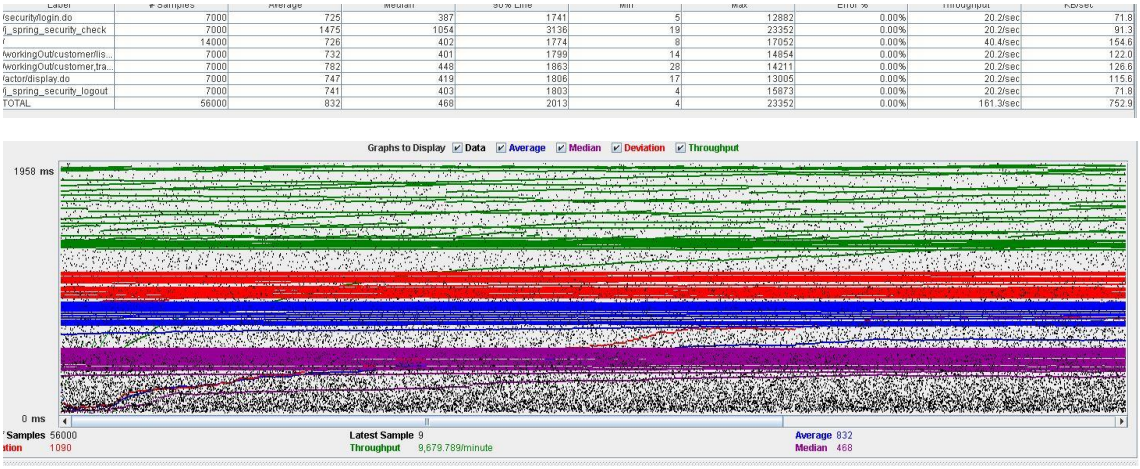
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	6500	560	247	1424	4	11255	0.00%	23.3/sec	82.7
/j_spring_security_check	6500	1131	731	2536	21	14208	0.00%	23.3/sec	105.3
/	13000	563	258	1402	7	12953	0.00%	46.5/sec	178.3
/workingOut/customer/lis...	6500	581	271	1404	14	15056	0.00%	23.3/sec	140.7
/workingOut/customer,tra...	6500	603	304	1418	24	17290	0.00%	23.3/sec	146.0
/actor/display.do	6500	578	276	1418	12	12884	0.00%	23.3/sec	133.2
/j_spring_security_logout	6500	587	276	1476	23	15473	0.00%	23.3/sec	82.7
TOTAL	52000	646	309	1598	4	17290	0.00%	186.1/sec	868.4



Overload test case: 140 concurrent users, 50 of loop count and 1 as ramp-up period.

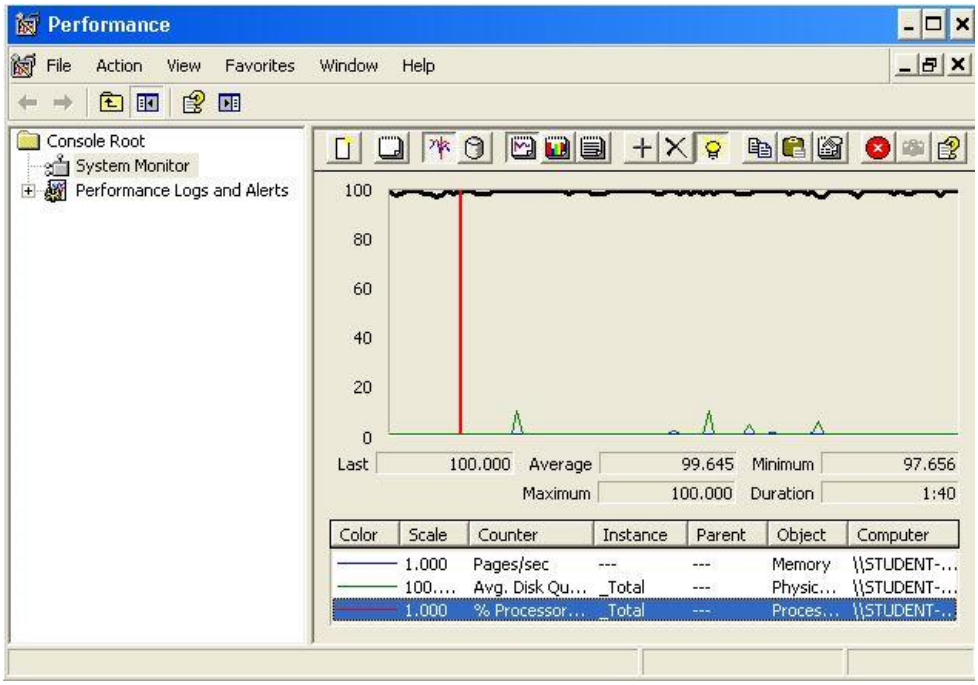


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	7000	725	387	1741	5	12882	0.00%	20.2/sec	71.8
/j_spring_security_check	7000	1475	1054	3136	19	23352	0.00%	20.2/sec	91.3
/	14000	726	402	1774	8	17052	0.00%	40.4/sec	154.6
/workingOut/customer/lis...	7000	732	401	1799	14	14854	0.00%	20.2/sec	122.0
/workingOut/customer,tra...	7000	782	448	1863	28	14211	0.00%	20.2/sec	126.6
/actor/display.do	7000	747	419	1806	17	13005	0.00%	20.2/sec	115.6
/j_spring_security_logout	7000	741	403	1803	4	15873	0.00%	20.2/sec	71.8
TOTAL	56000	832	468	2013	4	23352	0.00%	161.3/sec	752.9

With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.



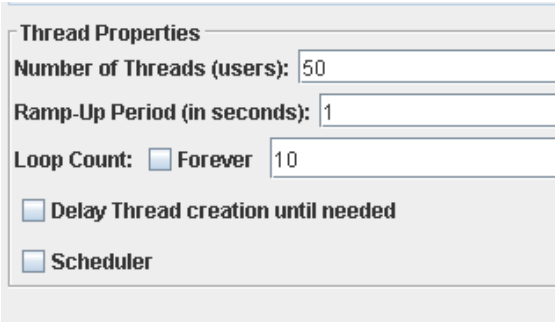
Conclusion: The maximum number of concurrent users supported by the system is 140 because the current CPU is a bottleneck.

Req. 10.3 - An actor who is authenticated as a customer must be able to manage his or her applications, which includes listing them, showing them, and creating them. When a customer applies for a working-out, he or she can add some comments.

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

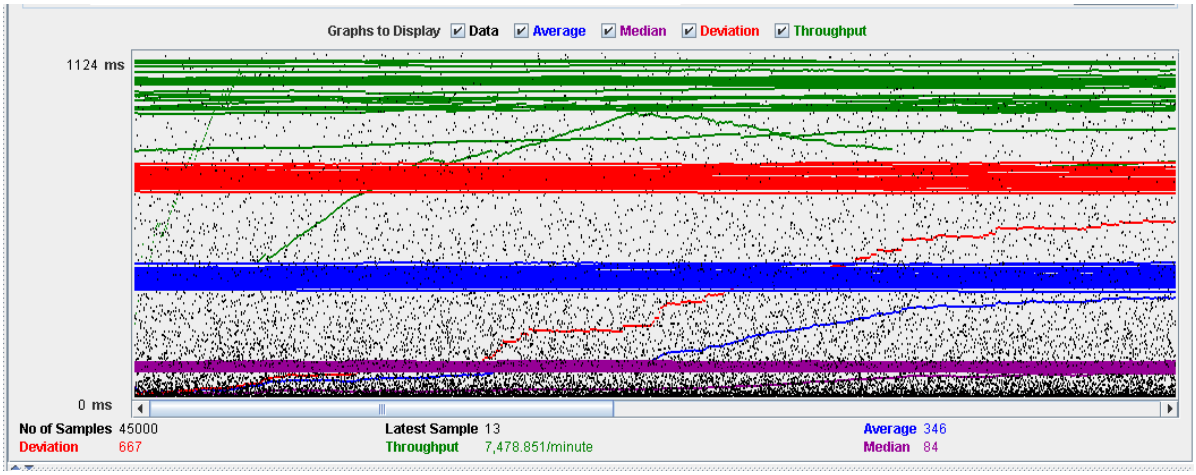
Test case description: The client user initiates a session, then lists the workouts, shows a working-outs, makes a application to that working-out. Attach your credit card in the working-out application and when saving the system redirects you to the list of applications. Finally show a application and close session.

Maximum workload test case: 50 concurrent users, 10 of loop count and 1 of ramp-up period.

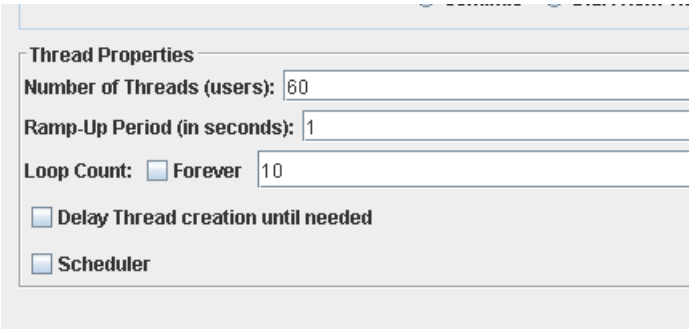


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	2500	392	160	1021	10	6366	0.00%	6.9/sec	24.7
/scripts/jquery.js	2500	32	28	56	6	253	0.00%	6.9/sec	1869.5
/styles/displayta...	2500	10	8	18	2	399	0.00%	6.9/sec	20.7
/scripts/jquery-ui...	2500	49	42	83	8	1261	0.00%	6.9/sec	3154.1
/styles/jmenu.css	2500	9	8	16	1	216	0.00%	6.9/sec	13.0
/scripts/md5-mi...	2500	9	7	17	2	155	0.00%	6.9/sec	37.5
/styles/common....	2500	9	7	16	2	185	0.00%	6.9/sec	4.0
/scripts/jmenu.js	2500	9	8	17	2	169	0.00%	6.9/sec	73.4
/j_spring_securit...	2500	837	491	1966	16	9774	0.00%	6.9/sec	31.4
/	5000	419	170	1074	10	8450	0.00%	13.9/sec	53.1
/workingOut/cust...	2500	440	206	1091	21	6424	0.00%	6.9/sec	42.0
/workingOut/cust...	2500	536	304	1238	35	6266	0.00%	6.9/sec	43.5
/application/cust...	2500	870	510	2051	24	9319	0.00%	6.9/sec	31.0
/application/cust...	2500	869	532	2016	26	9731	0.00%	6.9/sec	31.0
/application/cust...	2500	474	231	1163	28	6491	0.00%	7.0/sec	41.7
/application/cust...	2500	416	186	1069	15	5842	0.00%	7.0/sec	32.6
/j_spring_securit...	2500	421	175	1054	5	8518	0.00%	7.0/sec	24.9
TOTAL	45000	346	84	986	1	9774	0.00%	124.6/sec	5513.9



Overload test case: 60 concurrent users, 10 of loop count and 1 as ramp-up period.

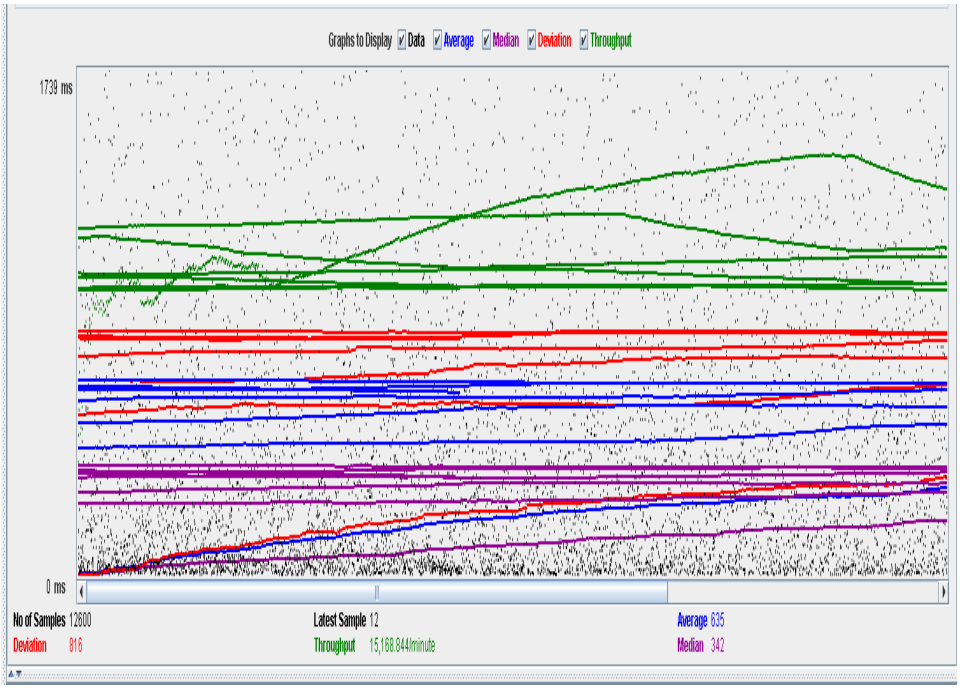


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1800	562	292	1477	4	7500	0.00%	36.2/sec	129.5
/j_spring_security_check	1800	1088	751	2475	15	6284	0.00%	36.2/sec	164.8
/	3600	556	283	1458	4	7423	0.00%	72.3/sec	278.0
/credit/CardCustomerList...	1800	563	273	1491	5	5578	0.00%	36.2/sec	187.9
/credit/CardCustomerIdis...	1800	557	290	1472	4	10510	0.00%	36.2/sec	164.7
/j_spring_security_logout	1800	560	296	1490	4	6147	0.00%	36.2/sec	129.6
TOTAL	12600	635	342	1654	4	10510	0.00%	252.8/sec	1052.6

With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.

In this analysis we can also see that hard disk make some peak. We think that is when the system goes to memory to get the list.



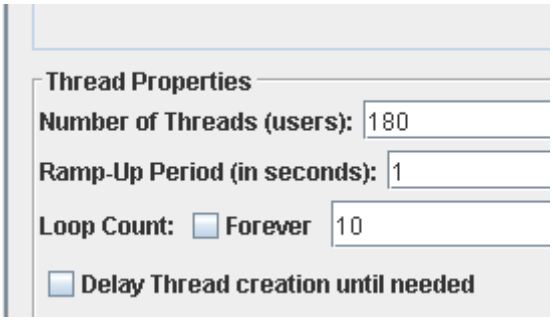
Conclusion: The maximum number of concurrent users supported by the system is 50 because the current CPU is a bottleneck.

Req. 10.4 - An actor who is authenticated as a customer must be able list and display their credit cards.

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

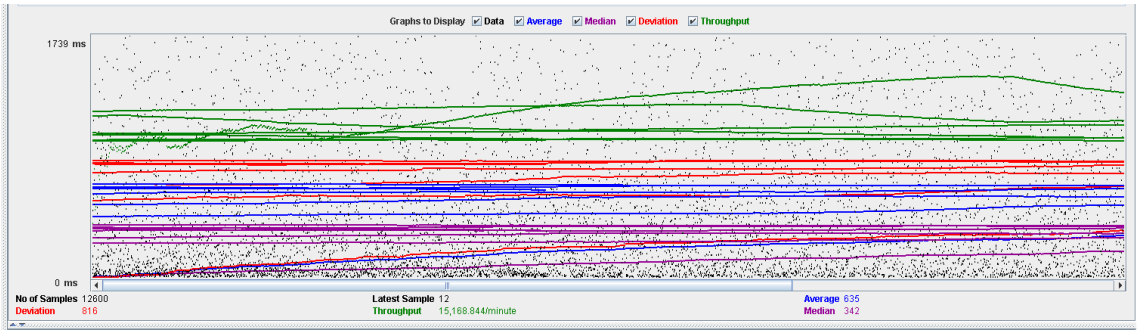
Test case description: The user login as a customer, go to credit cards list and display a credit card. Finally, the user logout.

Maximum workload test case: 180 concurrent users, 10 of loop count and 1 of ramp-up period.

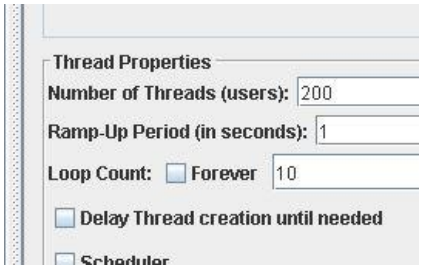


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

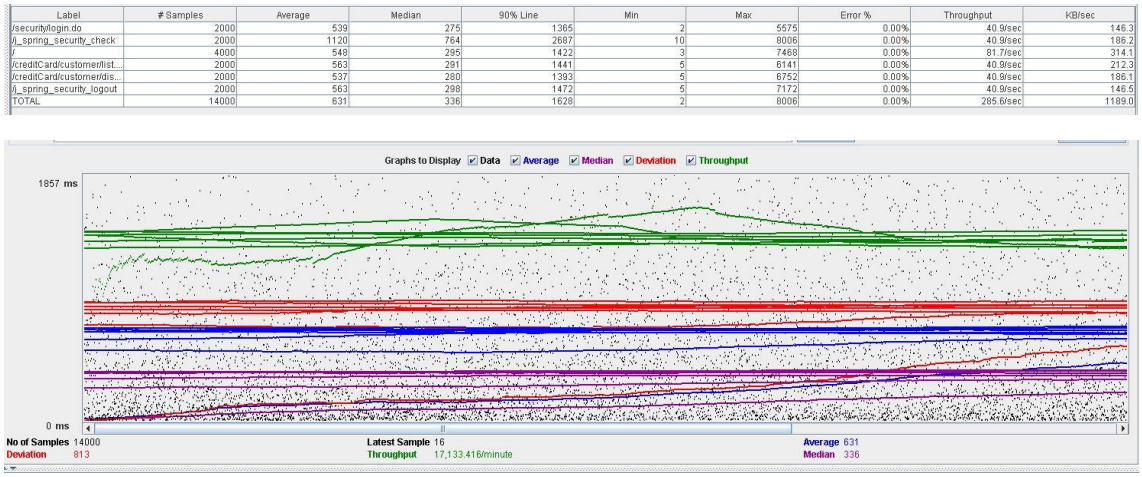
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1800	562	292	1477	4	7500	0.00%	36.2/sec	129.5
/j_spring_security_check	1800	1088	751	2475	15	6284	0.00%	36.2/sec	184.8
/	3600	556	283	1458	4	7423	0.00%	72.3/sec	278.0
/creditCard/customer/list...	1800	563	273	1491	5	5578	0.00%	36.2/sec	187.9
/creditCard/customer/dis...	1800	557	290	1472	4	10510	0.00%	36.2/sec	164.7
/j_spring_security_logout	1800	560	296	1490	4	6147	0.00%	36.2/sec	129.6
TOTAL	12600	635	342	1654	4	10510	0.00%	252.8/sec	1052.6



Overload test case: 200 concurrent users, 10 of loop count and 1 as ramp-up period.

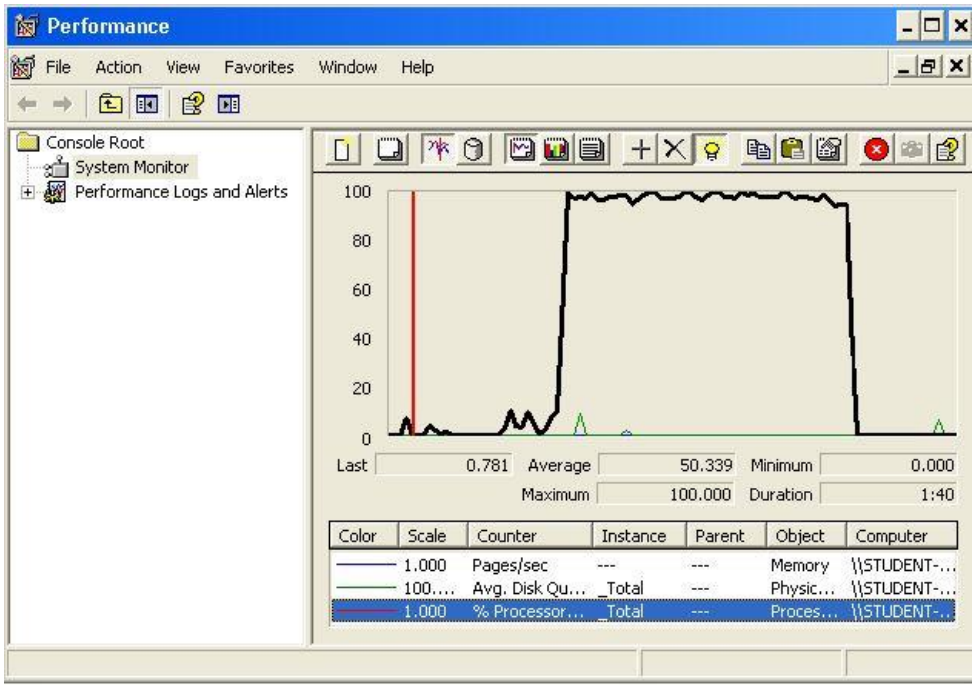


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	2000	539	275	1365	2	5575	0.00%	40.9/sec	146.3
/j_spring_security_check	2000	1120	764	2687	10	8006	0.00%	40.9/sec	186.2
/	4000	548	295	1422	3	7468	0.00%	81.7/sec	314.1
/creditCard/customer/list....	2000	563	291	1441	5	6141	0.00%	40.9/sec	212.3
/creditCard/customer/dis...	2000	537	280	1393	5	6752	0.00%	40.9/sec	186.1
/j_spring_security_logout	2000	563	298	1472	5	7172	0.00%	40.9/sec	146.6
TOTAL	14000	631	336	1628	2	8006	0.00%	205.6/sec	1189.0

With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.



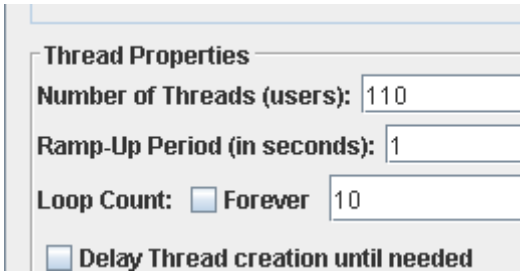
Conclusion: The maximum number of concurrent users supported by the system is 180 because the current CPU is a bottleneck.

Req. 10.4 - An actor who is authenticated as a customer must be able create and delete their credit cards.

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

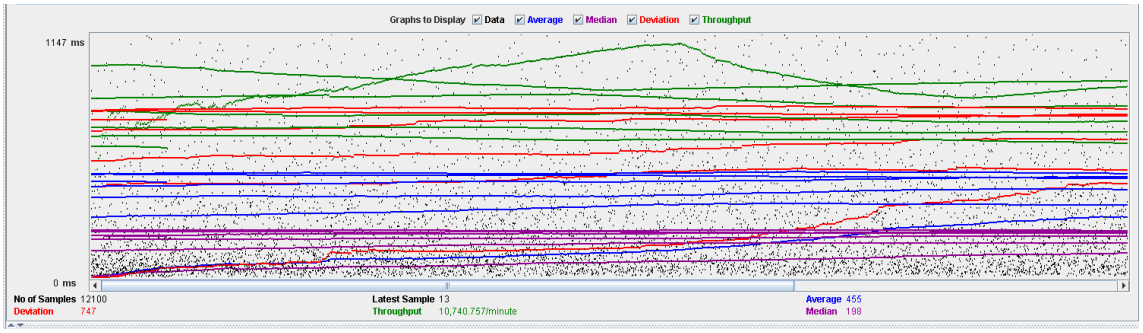
Test case description: The user login as a customer, go to credit cards create and delete a credit card. Finally, the user logout.

Maximum workload test case: 110 concurrent users, 10 of loop count and 1 of ramp-up period.

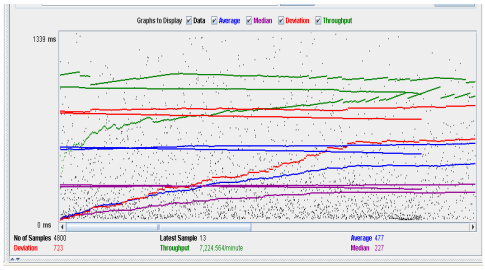


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1100	347	130	887	5	5571	0.00%	16.3/sec	58.2
/j_spring_security_check	1100	759	388	1855	13	9422	0.00%	16.3/sec	73.9
/	2200	343	133	829	5	8517	0.00%	32.6/sec	124.7
/creditCard/customer/list...	3300	351	142	839	7	8883	0.00%	48.9/sec	314.7
/creditCard/customer/cre...	1100	383	155	933	11	9000	0.00%	16.3/sec	99.0
/creditCard/customer/edi...	1100	710	418	1616	35	10064	0.00%	16.3/sec	114.3
/creditCard/customer/del...	1100	700	404	1600	20	8126	0.00%	16.3/sec	107.6
/j_spring_security_logout	1100	361	154	841	22	9159	0.00%	16.4/sec	58.1
TOTAL	12100	455	198	1112	5	10064	0.00%	179.0/sec	948.0



Overload test case: 125 concurrent users, 10 of loop count and 1 as ramp-up period.



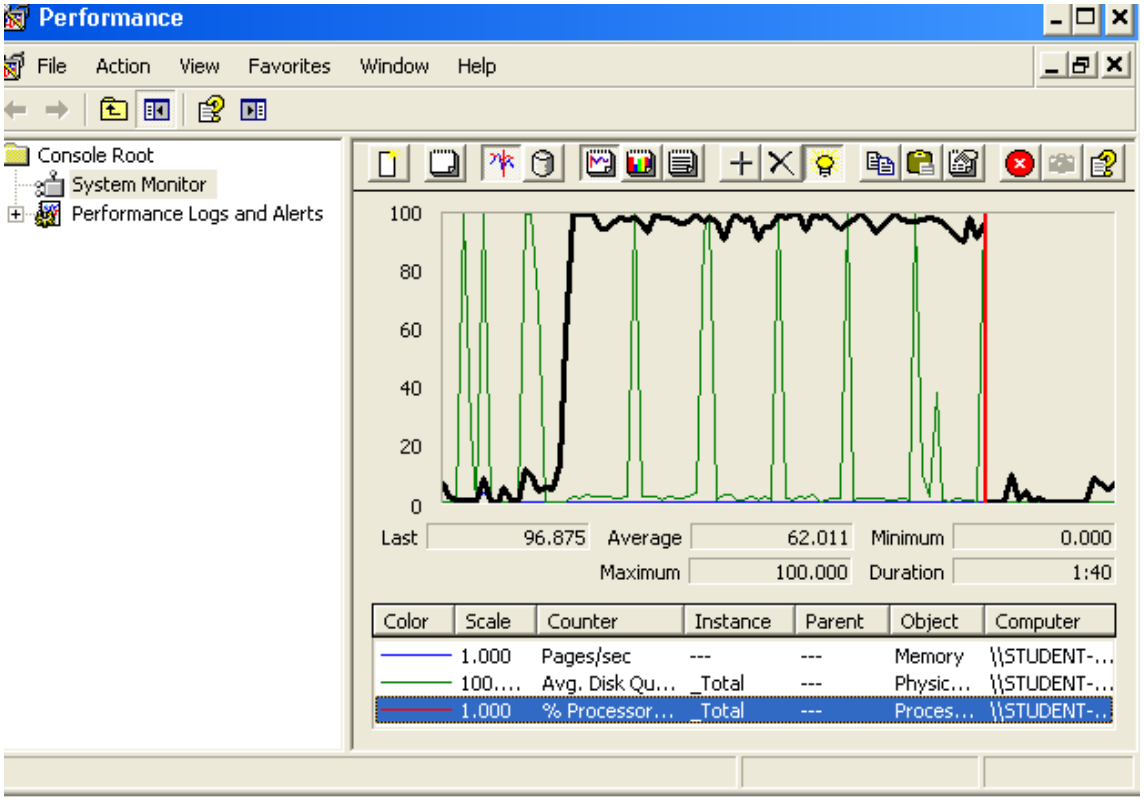
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

There are two methods that are implemented by us and have 90% line not acceptable, so we are going to study if it is possible to refactor this methods.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1250	651	225	1619	5	22173	0.00%	12.4/sec	44.2
/j_spring_security_check	1250	1075	637	2525	18	18279	0.00%	12.4/sec	56.1
/	2500	534	231	1282	5	22591	0.00%	24.8/sec	94.8
/creditCard/customer/list...	3750	590	245	1355	9	17107	0.00%	37.1/sec	239.2
/creditCard/customer/cre...	1250	592	236	1433	9	17438	0.00%	12.4/sec	75.1
/creditCard/customer/edi...	1250	1159	706	2481	46	18043	0.00%	12.4/sec	86.9
/creditCard/customer/del...	1250	1206	654	2688	44	19371	0.00%	12.4/sec	81.8
/j_spring_security_logout	1250	562	231	1325	17	17116	0.00%	12.4/sec	44.1
TOTAL	13750	735	332	1749	5	22591	0.00%	136.1/sec	720.8

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	800	391	169	993	9	5364	0.00%	20.1/sec	71.7
/j_spring_securit...	800	809	476	1796	14	7304	0.00%	20.1/sec	86.9
/	1600	378	161	938	10	6014	0.00%	40.2/sec	149.9
/dashboard/adm...	800	512	265	1133	35	5805	0.00%	20.1/sec	150.2
/j_spring_securit...	800	392	172	928	13	7202	0.00%	20.2/sec	71.6
TOTAL	4800	477	227	1159	9	7304	0.00%	120.4/sec	528.7

With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.



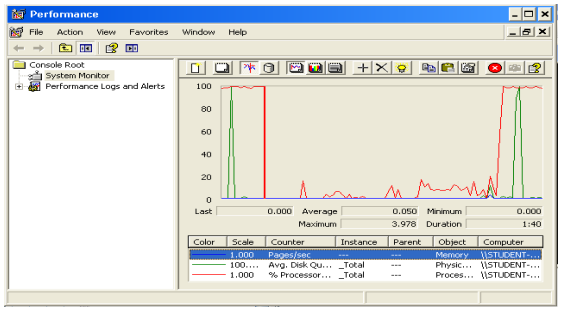
Conclusion: The maximum number of concurrent users supported by the system is 180 because the current CPU is a bottleneck.

Req. 11.4 – An actor who is authenticated as an administrator must be able to display a dashboard with the following information…

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

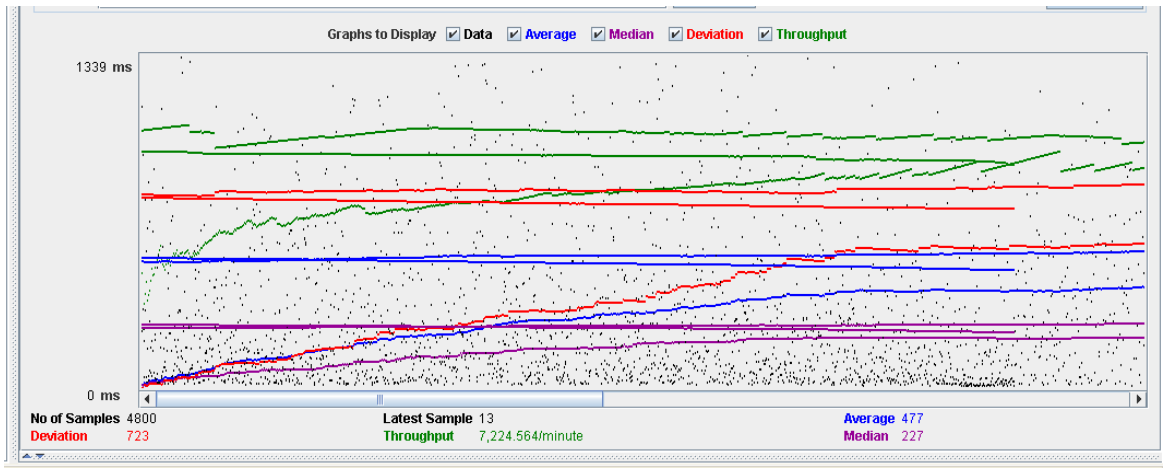
Test case description: The user admin starts session, then display dashboard and close session.

Maximum workload test case: 80 concurrent users, 10 of loop count and 1 of ramp-up period.

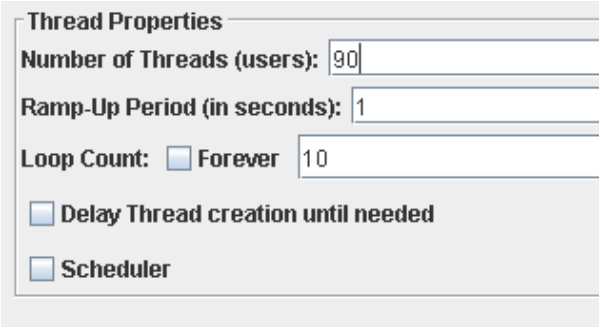


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	800	391	169	993	9	5364	0.00%	20.1/sec	71.7
/j_spring_securit...	800	809	476	1796	14	7304	0.00%	20.1/sec	86.9
/	1600	378	161	938	10	6014	0.00%	40.2/sec	149.9
/dashboard/adm...	800	512	265	1133	35	5805	0.00%	20.1/sec	150.2
/j_spring_securit...	800	392	172	928	13	7202	0.00%	20.2/sec	71.6
TOTAL	4800	477	227	1159	9	7304	0.00%	120.4/sec	528.7

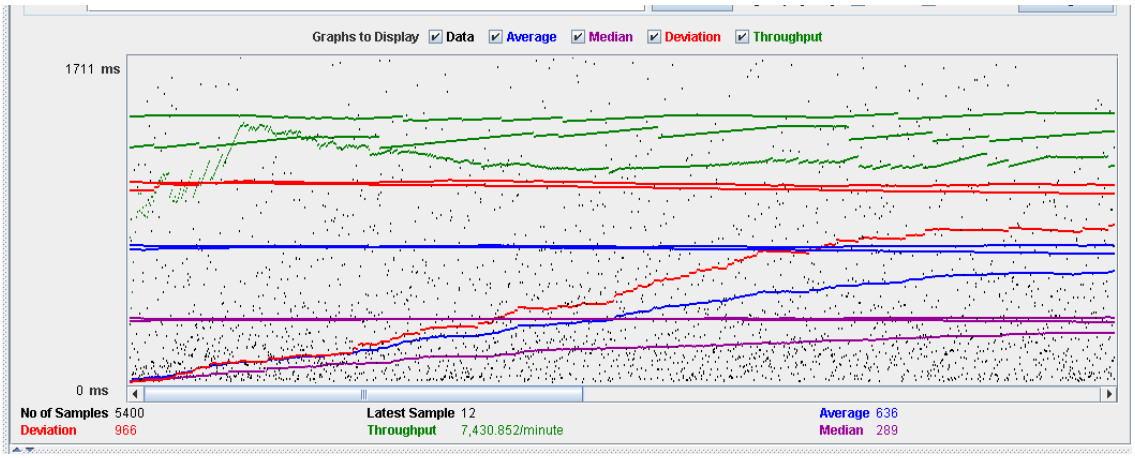


Overload test case: 90 concurrent users, 10 of loop count and 1 as ramp-up period.



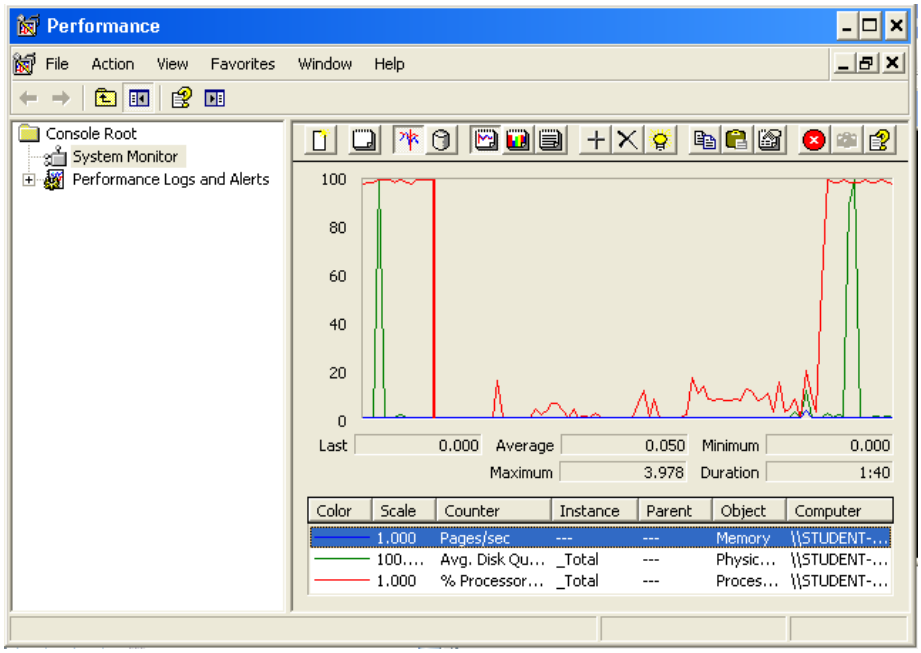
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	900	493	193	1212	10	8104	0.00%	20.7/sec	73.7
/j_spring_securit...	900	1021	562	2502	28	10567	0.00%	20.7/sec	89.4
/	1800	574	225	1483	12	9764	0.00%	41.4/sec	154.2
/dashboard/adm...	900	615	320	1351	40	8939	0.00%	20.7/sec	154.5
/j_spring_securit...	900	538	244	1289	5	6868	0.00%	20.7/sec	73.7
TOTAL	5400	636	289	1575	5	10567	0.00%	123.8/sec	543.8



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.

In this analysis we can also see that hard disk make some peak. We think that is when the system goes to memory to get the list.



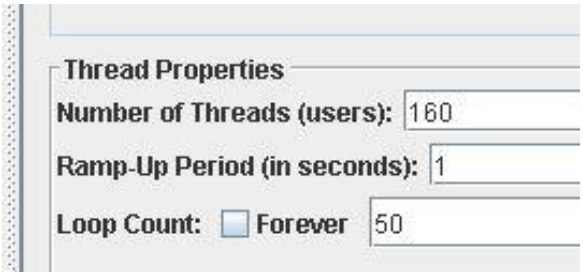
Conclusion: The maximum number of concurrent users supported by the system is 80 because the current CPU is a bottleneck.

Req. 13 – The system must be able to easy to customise at run time.

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

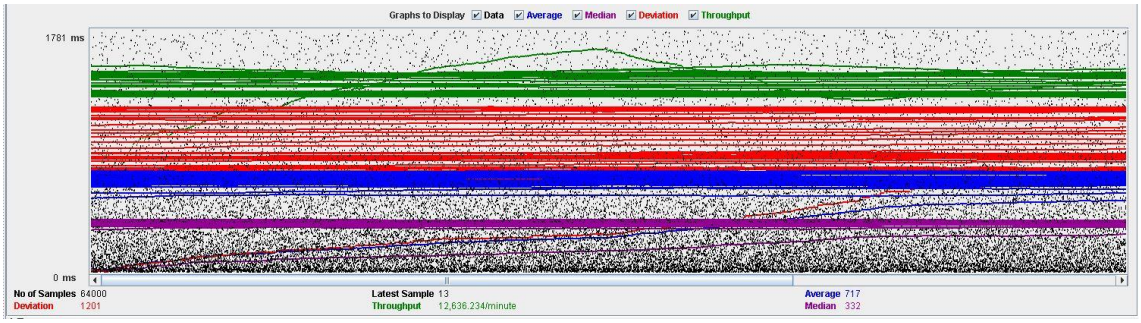
Test case description: The user login as an admin, go to customisation and edit link. He/she edits a value and save the form. Finally, the user logout.

Maximum workload test case: 160 concurrent users, 50 of loop count and 1 of ramp-up period.

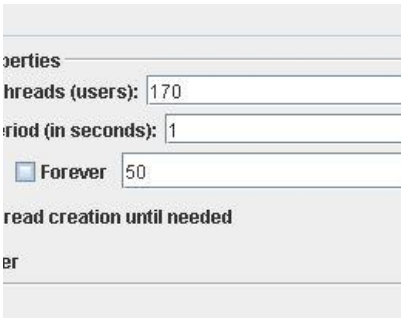


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/j_spring_security_check	8000	1115	638	2501	8	21515	0.16%	26.3/sec	114.5
/	16000	562	242	1348	3	19018	0.11%	52.7/sec	197.1
/customisation/administr...	16000	582	254	1387	4	20481	0.11%	52.7/sec	390.4
/customisation/administr...	16000	870	441	2078	5	20099	0.12%	52.7/sec	429.9
/j_spring_security_logout	8000	591	256	1392	5	19595	0.11%	26.3/sec	94.4
TOTAL	64000	717	332	1737	3	21515	0.12%	210.6/sec	1225.7



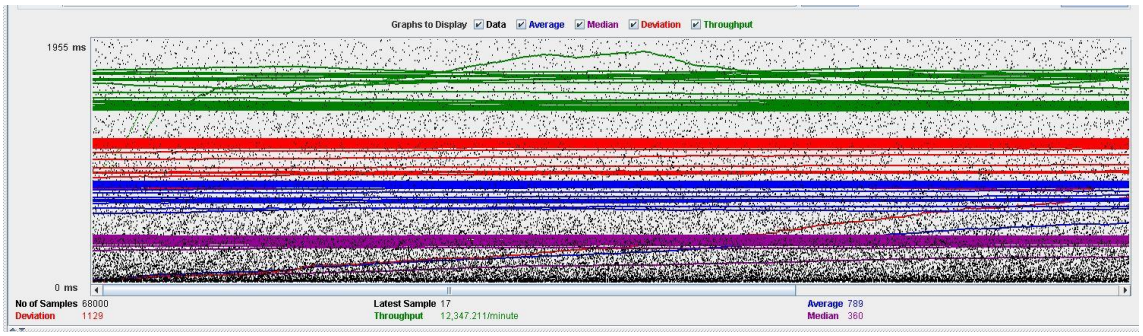
Overload test case: 170 concurrent users, 50 of loop count and 1 as ramp-up period.



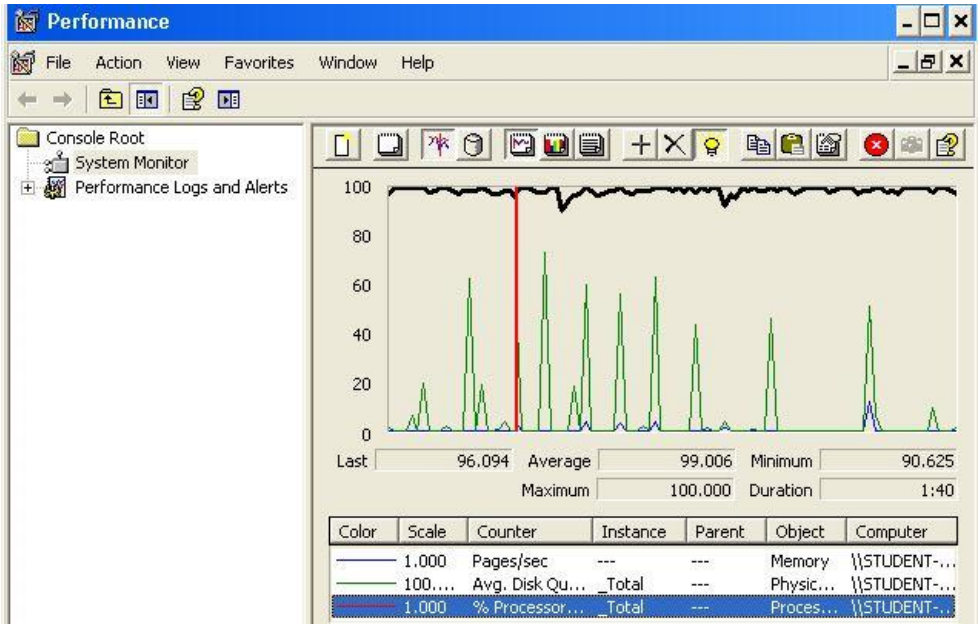
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Although when the customization is saving in the database and it is checking has a high 90% line. But it is normal, due to in this methods have to check a lot of attributes.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/j_spring_security_check	8500	1239	777	2960	9	16617	0.00%	25.7/sec	112.0
/	17000	631	251	1709	2	12603	0.00%	51.5/sec	192.7
/customisation/administr...	17000	631	262	1718	4	14872	0.00%	51.5/sec	381.9
/customisation/administr...	17000	950	492	2498	5	14178	0.00%	51.5/sec	420.5
/j_spring_security_logout	8500	645	269	1741	5	12111	0.00%	25.7/sec	92.3
TOTAL	68000	789	360	2124	2	16617	0.00%	205.8/sec	1199.0



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.



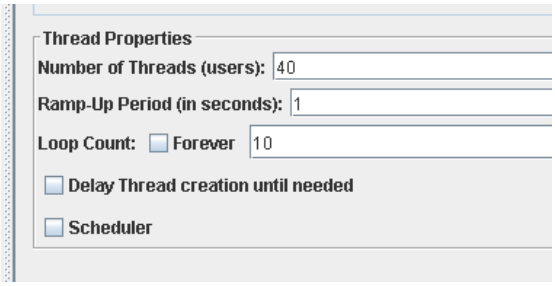
Conclusion: The maximum number of concurrent users supported by the system is 170 because the current CPU is a bottleneck.

Req. 46.1 – An actor who is authenticated as a nutritionist must be able to Manage his or her articles, which includes listing them, showing them, creating, updating and deleting them. An article can be saved in draft mode; once they are saved in final mode, they can't be edited.

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

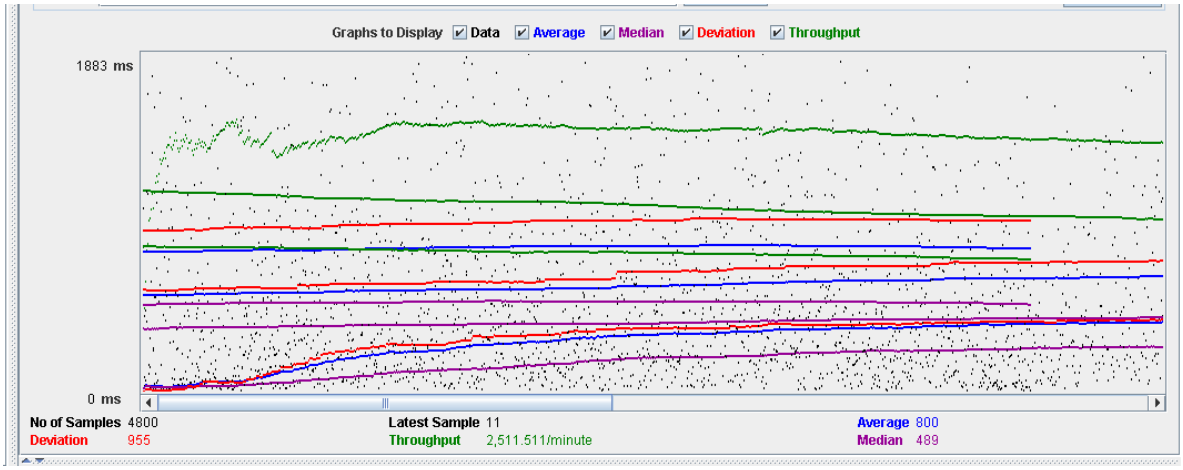
Test case description: The user nutritionist starts session, then lists the articles, creates a new article and then eliminates it. Finally, close session.

Maximum workload test case: 40 concurrent users, 10 of loop count and 1 of ramp-up period.

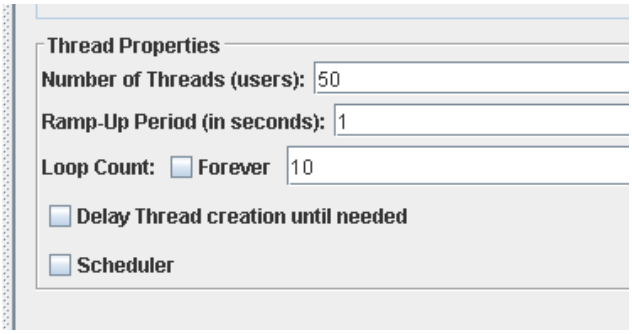


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	400	584	292	1556	10	7484	0.00%	3.5/sec	12.6
/j_spring_securit...	400	1039	715	2330	20	8609	0.00%	3.5/sec	13.4
/	800	588	290	1394	9	9800	0.00%	7.0/sec	24.2
/article/nutritioni...	1200	675	415	1471	22	8902	0.00%	10.5/sec	65.1
/article/nutritioni...	400	636	335	1506	16	10051	0.00%	3.5/sec	16.3
/article/nutritioni...	1200	1175	844	2503	16	7635	0.00%	10.5/sec	49.5
/j_spring_securit...	400	613	336	1503	16	6901	0.00%	3.5/sec	12.5
TOTAL	4800	800	489	1864	9	10051	0.00%	41.9/sec	192.6

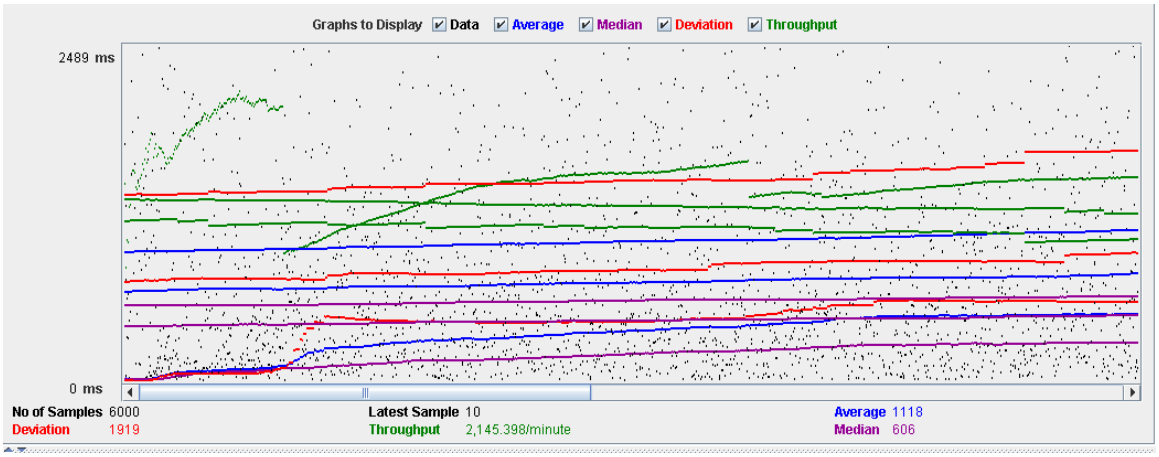


Overload test case: 50 concurrent users, 10 of loop count and 1 as ramp-up period.



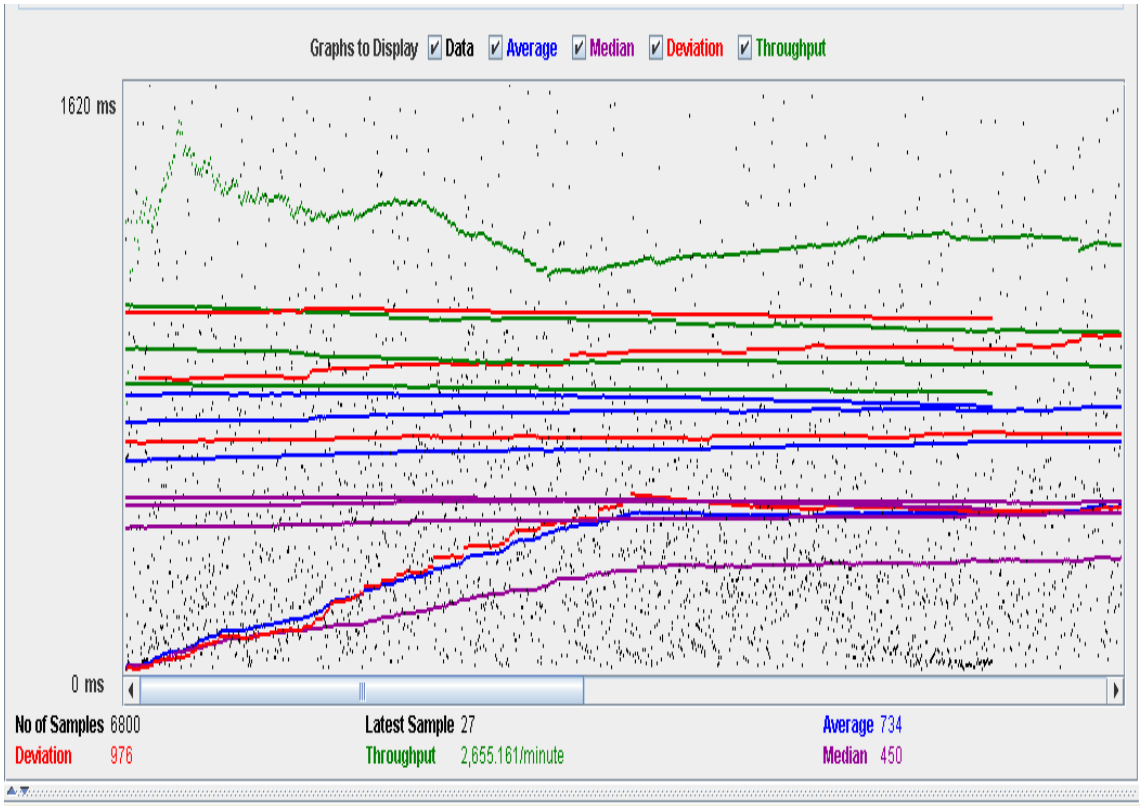
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	500	798	353	1784	9	10565	0.00%	3.0/sec	10.7
/j_spring_securit...	500	1414	978	2996	17	16557	0.00%	3.0/sec	11.4
/	1000	733	398	1829	10	12376	0.00%	6.0/sec	20.7
/article/nutritioni...	1500	968	550	2004	15	19525	0.00%	8.9/sec	55.6
/article/nutritioni...	500	852	392	1751	14	38441	0.00%	3.0/sec	13.8
/article/nutritioni...	1500	1713	1058	3418	21	45564	0.00%	9.0/sec	42.2
/j_spring_securit...	500	841	411	1927	12	15342	0.00%	3.0/sec	10.6
TOTAL	6000	1118	606	2484	9	45564	0.00%	35.8/sec	164.6



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.

In this analysis we can also see that hard disk make some peak. We think that is when the system goes to memory to get the list.



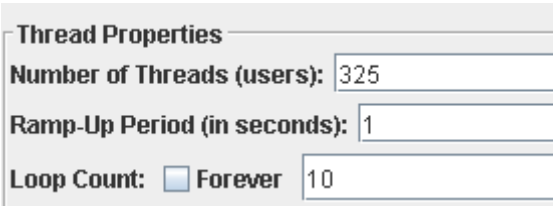
Conclusion: The maximum number of concurrent users supported by the system is 40 because the current CPU is a bottleneck.

Req. 34.3 - An actor who is authenticated as a trainer must be able to: Manage his or her endorsement, which include listing, showing, creating, updating and deleting them.

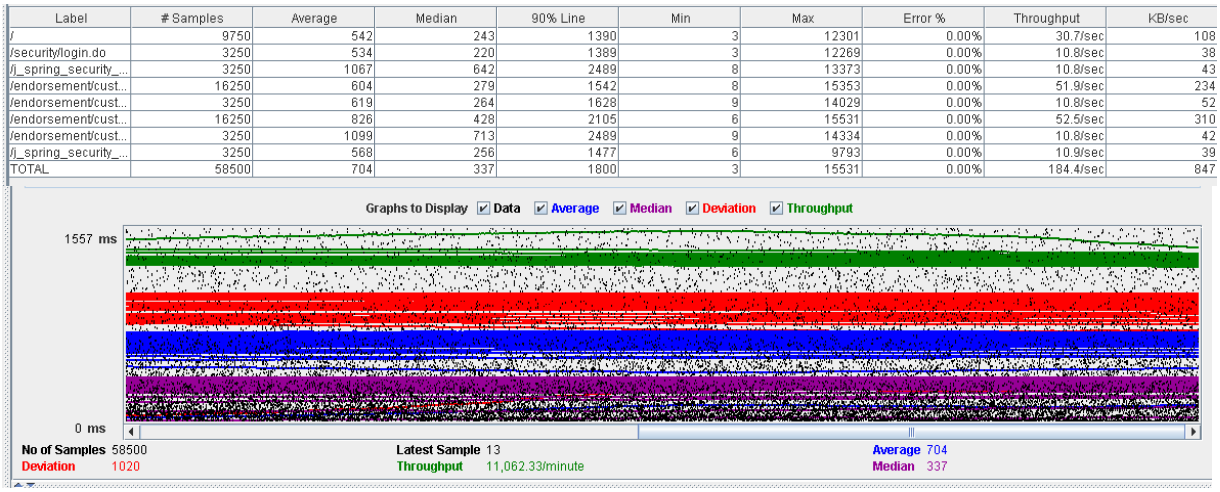
- RAM: 8 GB
- CPU: Intel Core i7-7700HQ
- Hard disk: 256 GB SSD + 921 GB HDD
- Network card: Intel(R) Dual Band Wireless-AC 3168

Test case description: The user logs in the system as a trainer, then, he/she clicks on *Endorsement list* option in the main menu to list the endorsements, then the user creates one and display it. Next, he/she updates one field and press Save. Ending, the trainer deletes it from the system and logs out.

Maximum workload test case: 325 concurrent users, 10 of loop count and 1 of ramp-up period.

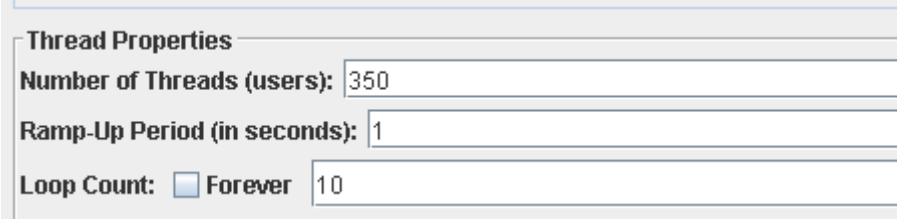


With this configuration we make sure that the system doesn't produce any errors and has a good response time.



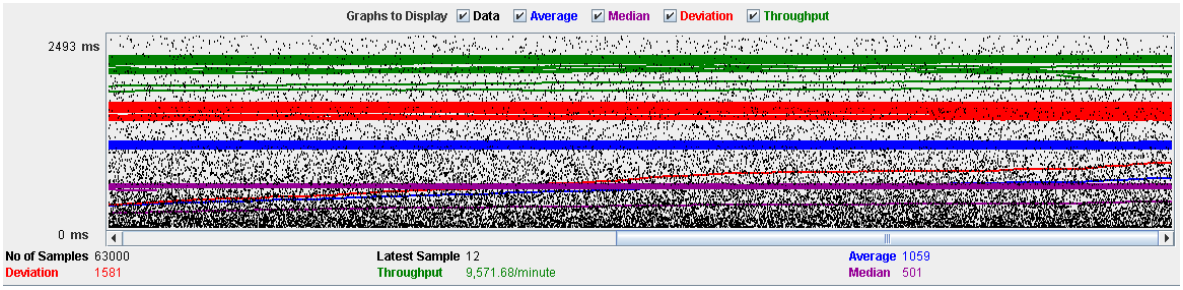
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	9750	542	243	1390	3	12301	0.00%	30.7/sec	108
/security/login.do	3250	534	220	1389	3	12269	0.00%	10.8/sec	38
/j_spring_security_...	3250	1067	642	2489	8	13373	0.00%	10.8/sec	43
/endorsement/cust...	16250	604	279	1542	8	15353	0.00%	51.9/sec	234
/endorsement/cust...	3250	619	264	1628	9	14029	0.00%	10.8/sec	52
/endorsement/cust...	16250	826	428	2105	6	15531	0.00%	52.5/sec	310
/endorsement/cust...	3250	1099	713	2489	9	14334	0.00%	10.8/sec	42
/j_spring_security_...	3250	568	256	1477	6	9793	0.00%	10.9/sec	39
TOTAL	58500	704	337	1800	3	15531	0.00%	184.4/sec	847

Overload test case: 350 concurrent users, 10 of loop count and 1 as ramp-up period.



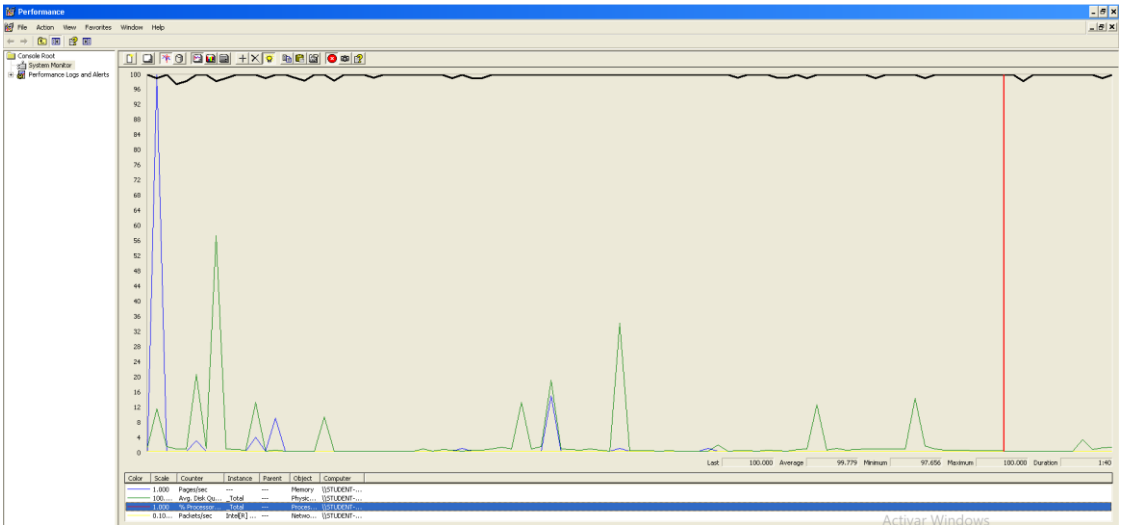
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time is "/endorsement/customer,trainer/delete.do". This method is an implementation of the developer team so it's possible to improve it refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	10500	842	383	2124	3	25493	0.00%	26.6/sec	93.9
/security/login.do	3500	786	348	1974	4	18750	0.00%	9.3/sec	33.0
/j_spring_security_...	3500	1704	1058	3816	9	18850	0.00%	9.3/sec	37.2
/endorsement/cust...	17500	908	414	2312	7	21630	0.00%	44.9/sec	203.3
/endorsement/cust...	3500	957	435	2414	8	19510	0.00%	9.3/sec	45.2
/endorsement/cust...	17500	1197	595	2981	6	32906	0.00%	45.3/sec	267.6
/endorsement/cust...	3500	1736	1083	4006	8	25778	0.00%	9.3/sec	36.9
/j_spring_security_...	3500	823	371	2048	6	14867	0.00%	9.5/sec	34.0
TOTAL	63000	1059	501	2683	3	32906	0.00%	159.5/sec	733.9



With the following picture is performance analysis in which we can see that CPU are working more than 95-100% all the time. So, if we improve the resources of the CPU and try to refactor the code of the method with the highest time is possible that the system supports more user concurrently.

In this analysis we can also see that hard disk make some peak. We think that is when the system goes to memory to get the list.



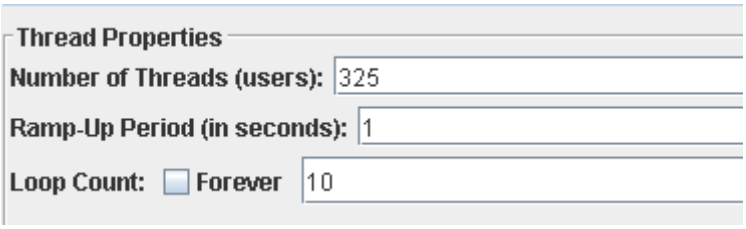
Conclusion: The maximum number of concurrent users supported by the system is 325 because the current CPU is a bottleneck.

Req. 36.3 - An actor who is authenticated as a customer must be able to: Manage his or her endorsement, which include listing, showing, creating, updating and deleting them.

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

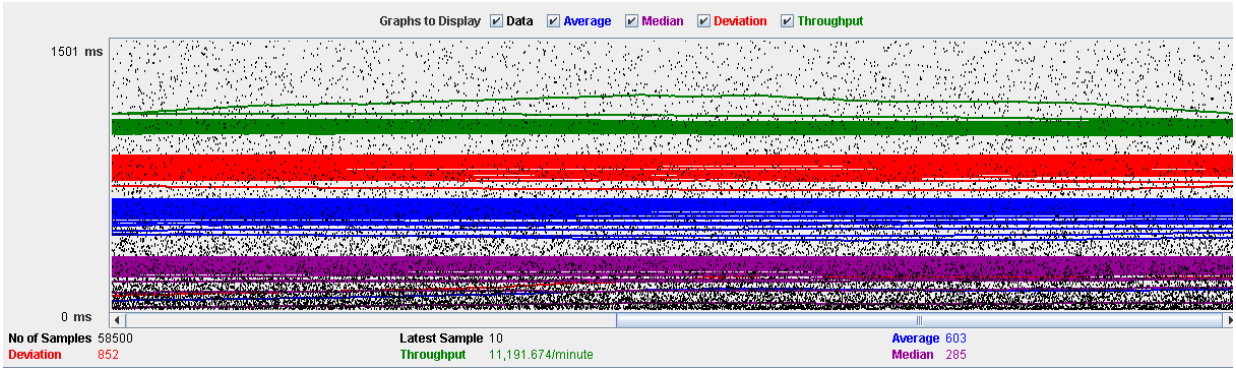
Test case description: The user logs in the system as a customer, then, he/she clicks on *Endorsement list* option in the main menu to list the endorsements, then the user creates one and display it. Next, he/she updates one field and press Save. Ending, the trainer deletes it from the system and logs out.

Maximum workload test case: 325 concurrent users, 10 of loop count and 1 of ramp-up period.

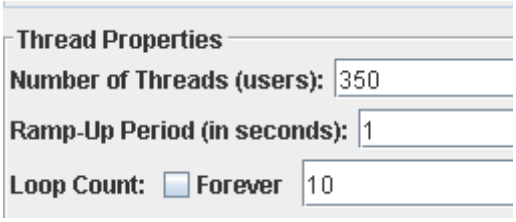


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	9750	441	187	1146	3	8973	0.00%	31.1/sec	115.3
/security/login.do	3250	433	174	1177	2	7909	0.00%	11.0/sec	39.3
/j_spring_security_c...	3250	891	535	2135	8	8605	0.00%	11.0/sec	50.2
/endorsement/custo...	16250	513	240	1338	7	9968	0.00%	52.8/sec	267.1
/endorsement/custo...	3250	541	236	1414	6	8214	0.00%	11.0/sec	59.5
/endorsement/custo...	16250	723	378	1850	5	10960	0.00%	53.4/sec	344.2
/endorsement/custo...	3250	1035	641	2491	8	9173	0.00%	11.0/sec	49.6
/j_spring_security_l...	3250	448	192	1179	6	7734	0.00%	11.1/sec	39.8
TOTAL	58500	603	285	1585	2	10960	0.00%	186.5/sec	935.2

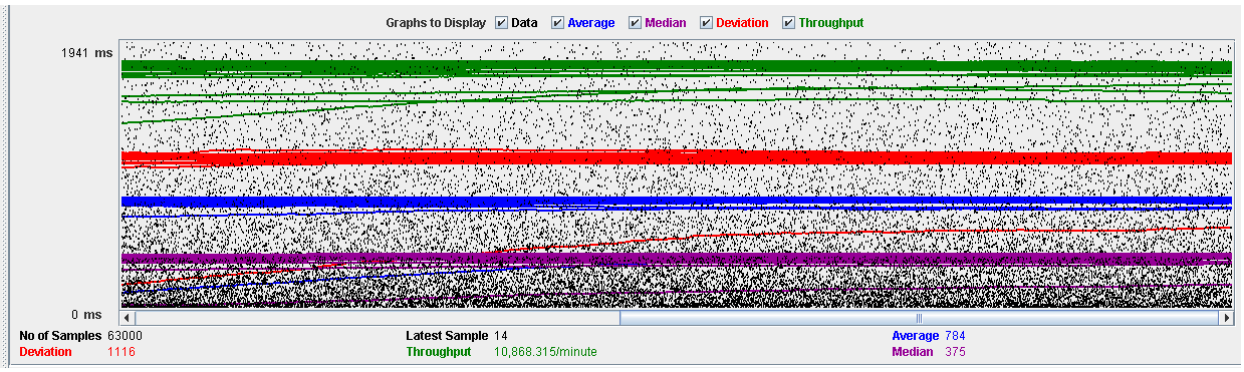


Overload test case: 350 concurrent users, 10 of loop count and 1 as ramp-up period.

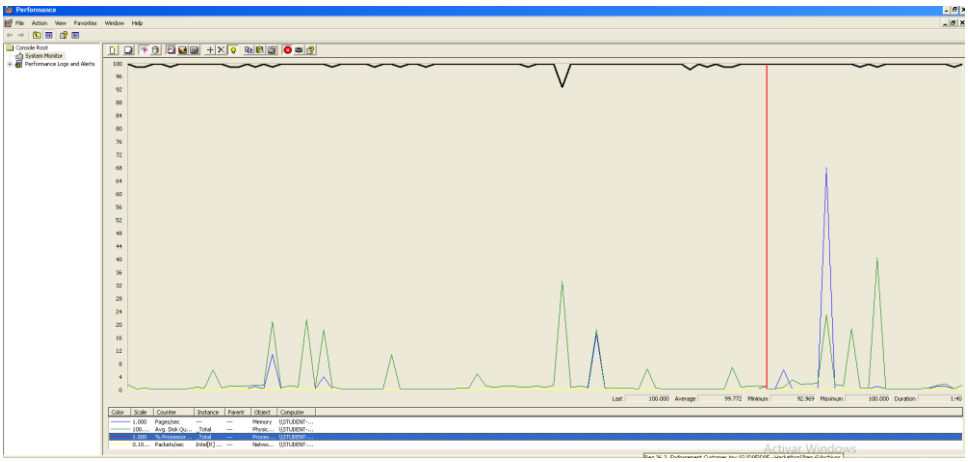


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time is "/endorsement/customer,trainer/delete.do". This method is an implementation of the developer team so it's possible to improve it refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	10500	589	262	1545	3	11236	0.00%	30.2/sec	112.(
/security/login.do	3500	590	250	1564	3	10175	0.00%	10.6/sec	38.(
/j_spring_security_c...	3500	1253	805	2892	7	14532	0.00%	10.7/sec	48.(
/endorsement/custo...	17500	661	304	1705	8	12191	0.00%	51.3/sec	259.(
/endorsement/custo...	3500	711	331	1787	8	14851	0.00%	10.7/sec	57.7
/endorsement/custo...	17500	906	463	2314	5	15977	0.00%	51.8/sec	334.(
/endorsement/custo...	3500	1328	859	3202	8	13676	0.00%	10.7/sec	48.2
/j_spring_security_l...	3500	629	276	1662	5	18900	0.00%	10.8/sec	38.(
TOTAL	63000	784	375	2021	3	18900	0.00%	181.1/sec	909.(



With the following picture is performance analysis in which we can see that CPU are working more than 95-100% all the time. So, if we improve the resources of the CPU and try to refactor the code of the method with the highest time is possible that the system supports more user concurrently.



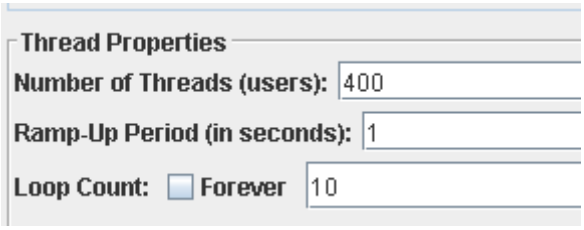
Conclusion: The maximum number of concurrent users supported by the system is 325 because the current CPU is a bottleneck.

Req. 36.4 - An actor who is authenticated as a customer must be able to: List and display the endorsement that other customer has written about the trainer he or she has attended his working-outs.

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

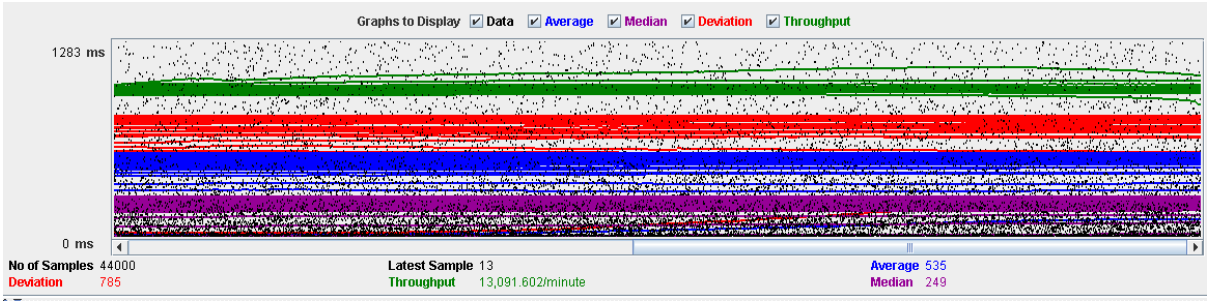
Test case description: The user logs in the system, goes to *Working-Out list* option in main menu, display a working out of a trainer attended, display the profile of the trainer, click on *Endorsement list* and display one.

Maximum workload test case: 400 concurrent users, 10 of loop count and 1 of ramp-up period.

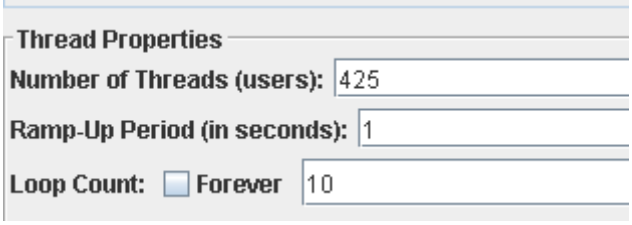


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	12000	458	200	1212	2	9539	0.00%	59.5/sec	220.
/security/login.do	4000	438	199	1170	2	7565	0.00%	21.1/sec	75.
/j_spring_security_...	4000	959	623	2219	6	10144	0.00%	21.0/sec	95.
/workingOut/custo...	4000	523	238	1342	7	11663	0.00%	21.1/sec	127.
/workingOut/custo...	4000	593	292	1491	12	9247	0.00%	21.1/sec	143.
/actor/display.do	4000	539	263	1402	6	11332	0.00%	21.1/sec	121.
/endorsement/cust...	4000	505	237	1341	6	6569	0.00%	21.1/sec	104.
/endorsement/cust...	4000	474	217	1208	4	8988	0.00%	21.1/sec	96.
/j_spring_security_...	4000	483	216	1253	4	8084	0.00%	21.2/sec	75.
TOTAL	44000	535	249	1385	2	11663	0.00%	218.2/sec	1011.

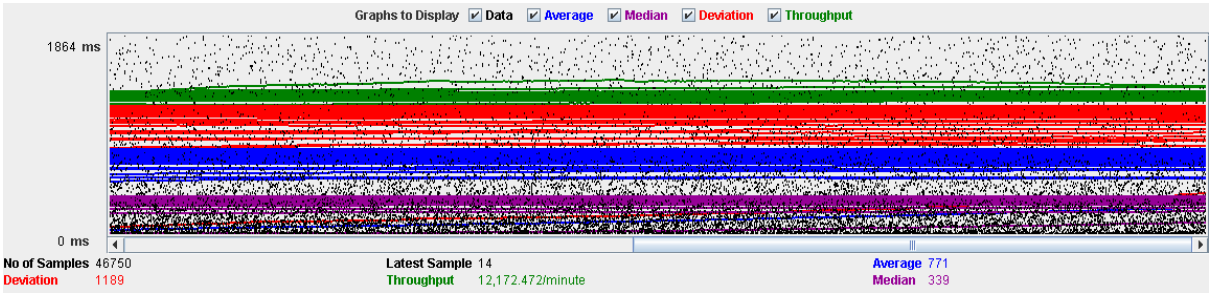


Overload test case: 425 concurrent users, 10 of loop count and 1 as ramp-up period.

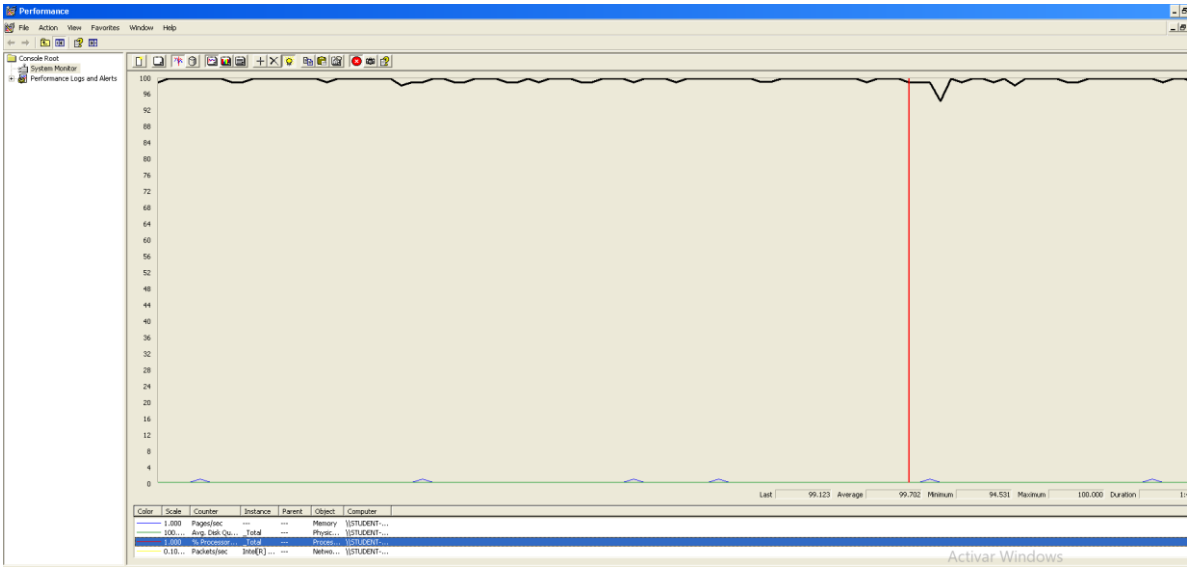


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time is "j_spring_security_check". This method is not an implementation of the developer team so it's not possible to improve it refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	12750	672	277	1781	3	18746	0.00%	55.3/sec	205.
/security/login.do	4250	682	276	1718	3	16652	0.00%	19.5/sec	69.
/j_spring_security_...	4250	1333	826	3138	7	13275	0.00%	19.5/sec	88.
/workingOut/custo...	4250	765	339	1994	6	13430	0.00%	19.5/sec	118.
/workingOut/custo...	4250	805	394	2072	12	11636	0.00%	19.5/sec	132.
/actor/display.do	4250	764	332	1965	6	14393	0.00%	19.5/sec	111.
/endorsement/cust...	4250	734	320	1893	5	12554	0.00%	19.5/sec	96.
/endorsement/cust...	4250	699	293	1833	4	12767	0.00%	19.5/sec	89.
/j_spring_security_...	4250	684	278	1750	6	20210	0.00%	19.6/sec	70.
TOTAL	46750	771	339	2031	3	20210	0.00%	202.9/sec	940.



With the following picture is performance analysis in which we can see that CPU are working more than 95-100% all the time. So, if we improve the resources of the CPU is possible that the system supports more user concurrently.



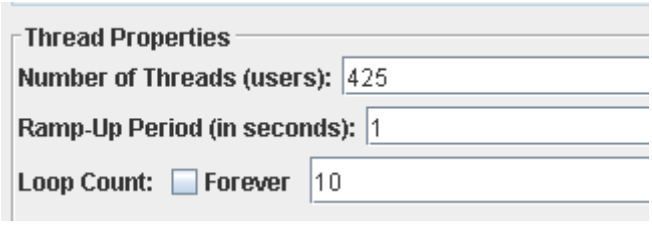
Conclusion: The maximum number of concurrent users supported by the system is 400 because the current CPU is a bottleneck.

Req. 37.1 - An actor who is authenticated as an administrator must be able to: Create an account for a new audit.

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

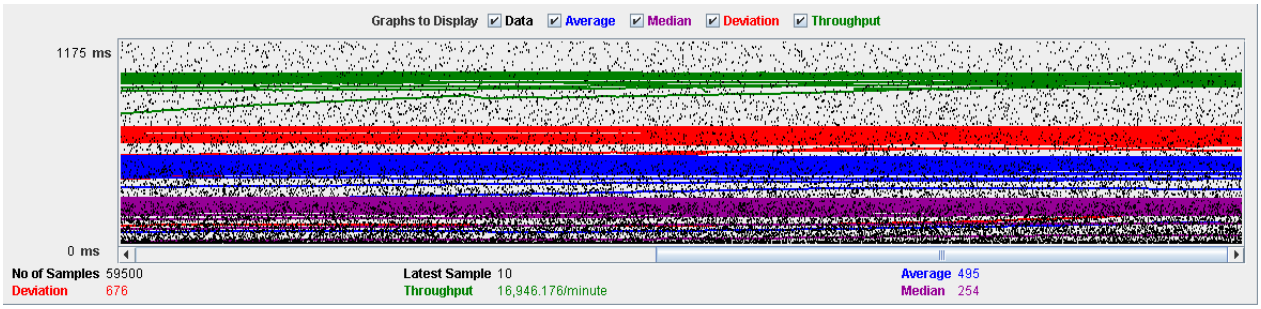
Test case description: The user logs in the system as an administrator, then, he/she clicks on *Register auditor* in the main menu, fills the form and save it. Finally, logs out and logs in the system with the new account.

Maximum workload test case: 425 concurrent users, 10 of loop count and 1 of ramp-up period.

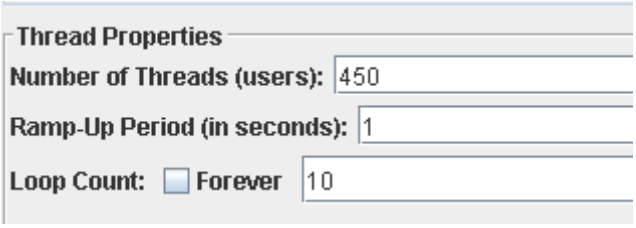


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	21250	421	206	1100	3	7619	0.00%	100.9/sec	360.6
/security/login.do	8500	410	202	1053	2	7297	0.00%	41.2/sec	147.2
/j_spring_security_...	8500	854	588	1952	5	10459	0.00%	41.2/sec	168.2
/actor/administrato...	8500	466	243	1193	3	6558	0.00%	42.5/sec	344.5
/welcome/index.do	4250	468	236	1229	3	6658	0.00%	21.4/sec	88.5
/j_spring_security_...	8500	452	228	1149	4	12584	0.00%	41.8/sec	150.1
TOTAL	59500	495	254	1280	2	12584	0.00%	282.4/sec	1224.6

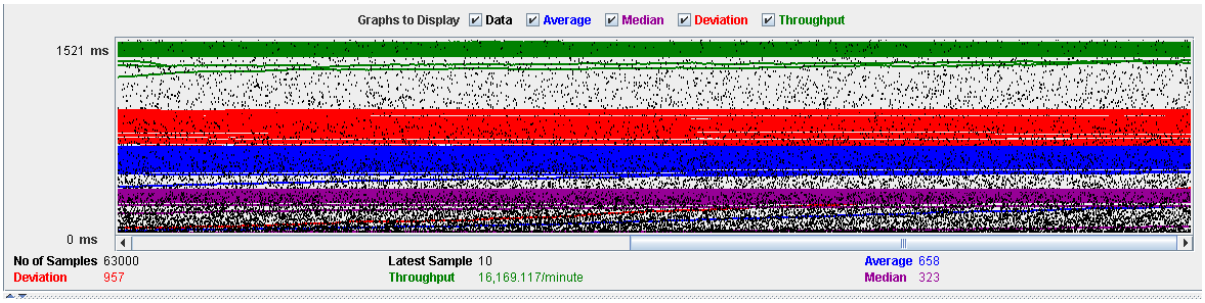


Overload test case: 450 concurrent users, 10 of loop count and 1 as ramp-up period.

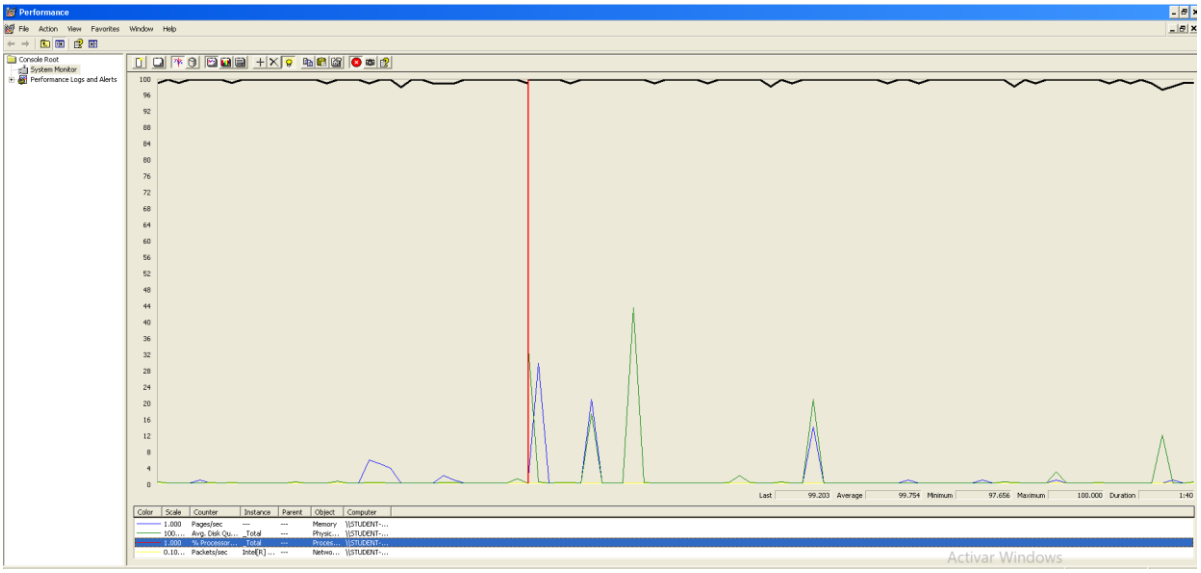


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time is "j_spring_security_check". This method is not an implementation of the developer team so it's not possible to improve it refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	22500	564	261	1428	3	16092	0.00%	96.2/sec	344.1
/security/login.do	9000	558	256	1429	2	11845	0.00%	39.2/sec	140.1
/j_spring_security_...	9000	1115	723	2558	6	17029	0.00%	39.2/sec	160.3
/actor/administrato...	9000	624	302	1569	5	14303	0.00%	40.2/sec	326.5
/welcome/index.do	4500	598	288	1522	3	8594	0.00%	20.3/sec	83.7
/j_spring_security_...	9000	600	306	1473	4	11141	0.00%	39.8/sec	142.8
TOTAL	63000	658	323	1651	2	17029	0.00%	269.5/sec	1168.7



With the following picture is performance analysis in which we can see that CPU are working more than 97-100% all the time. So, if we improve the resources of the CPU is possible that the system supports more user concurrently.



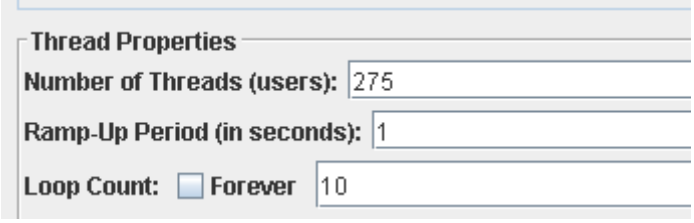
Conclusion: The maximum number of concurrent users supported by the system is 425 because the current CPU is a bottleneck.

Req. 37.2 - An actor who is authenticated as an administrator must be able to: Display a listing of actors including suspicious ones, ban an actor who is considered suspicious and unban an actor.

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

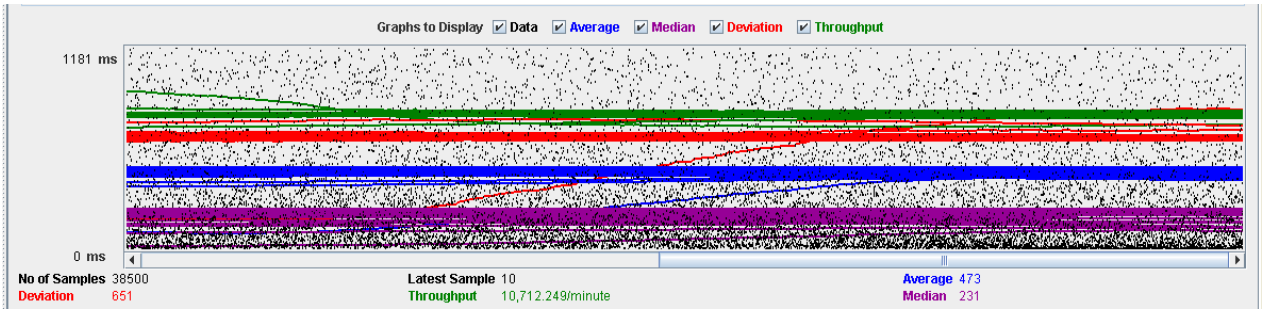
Test case description: The user logs in the system as an administrator, then, he/she clicks on *Actor list* in the main menu, launch suspicious actor process, display the actor who is considered suspicious and ban him/her, then unban and logs out the system.

Maximum workload test case: 275 concurrent users, 10 of loop count and 1 of ramp-up period.

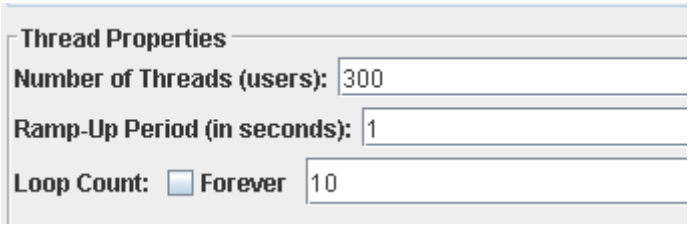


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	2750	262	91	712	2	5755	0.00%	13.5/sec	49.0
/j_spring_security_...	2750	577	331	1393	5	7917	0.00%	13.5/sec	58.5
/	5500	290	115	762	3	6294	0.00%	25.7/sec	96.0
/actor/administrato...	8250	437	214	1101	7	7520	0.00%	39.8/sec	258.0
/actor/administrato...	2750	949	630	2215	20	8174	0.00%	13.5/sec	91.8
/actor/display.do	8250	392	190	1020	6	7386	0.00%	40.0/sec	215.3
/actor/administrato...	2750	765	526	1755	12	7630	0.00%	13.6/sec	75.5
/actor/administrato...	2750	693	458	1610	10	5516	0.00%	13.6/sec	65.6
/j_spring_security_...	2750	312	137	808	4	4306	0.00%	13.6/sec	48.6
TOTAL	38500	473	231	1235	2	8174	0.00%	178.5/sec	916.2

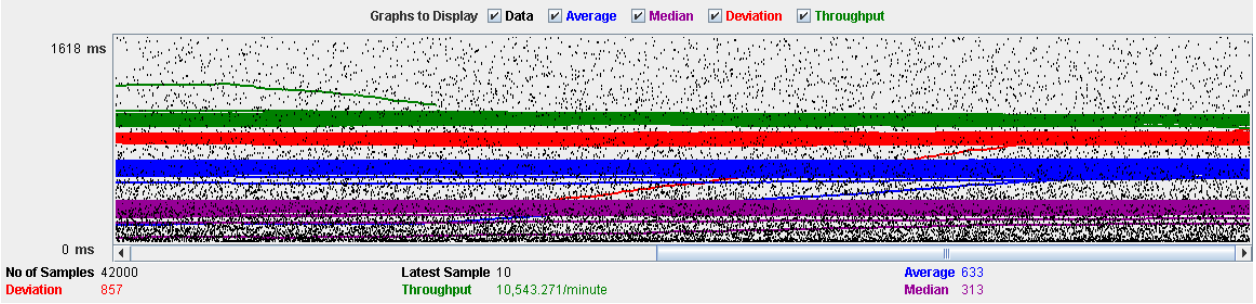


Overload test case: 300 concurrent users, 10 of loop count and 1 as ramp-up period.

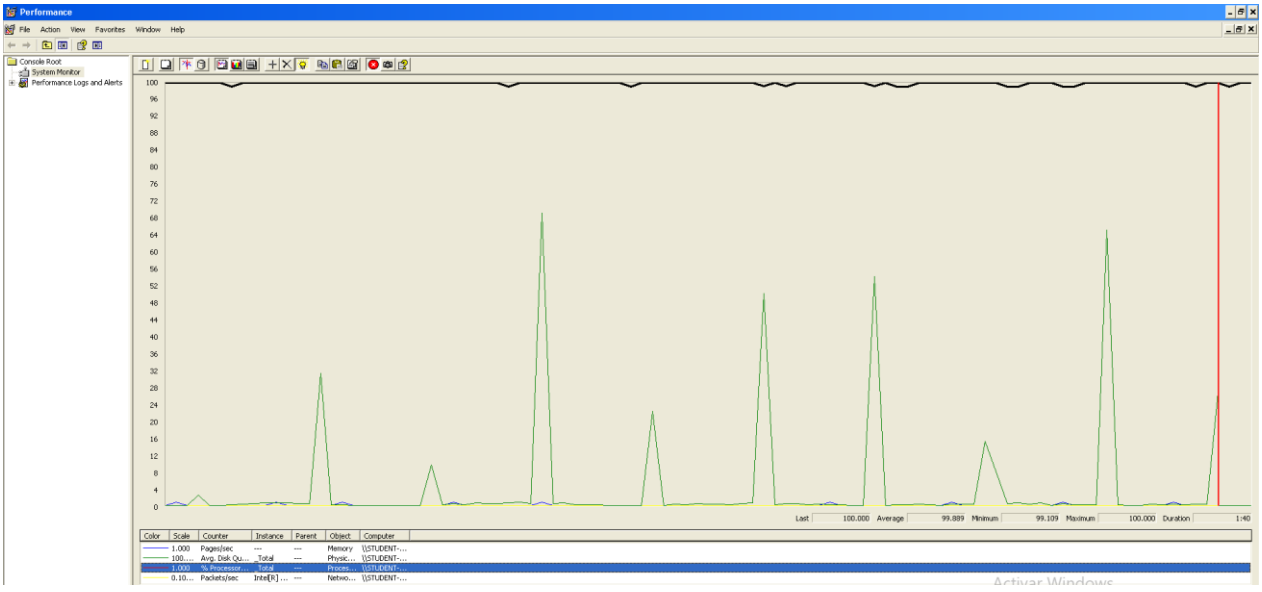


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time is "/actor/administrator/suspiciousProcess.do". This method is an implementation of the developer team so it's possible to improve it refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	3000	400	160	1072	2	6243	0.00%	13.2/sec	48.1
/j_spring_security_...	3000	871	537	2118	6	8863	0.00%	13.2/sec	57.4
/	6000	439	193	1203	3	7701	0.00%	25.3/sec	94.5
/actor/administrato...	9000	544	263	1409	7	9059	0.00%	39.1/sec	253.3
/actor/administrato...	3000	1190	806	2802	21	8089	0.00%	13.2/sec	90.0
/actor/display.do	9000	489	237	1278	6	7893	0.00%	39.2/sec	211.6
/actor/administrato...	3000	1007	657	2377	13	10477	0.00%	13.3/sec	74.2
/actor/administrato...	3000	975	647	2217	9	11736	0.00%	13.3/sec	64.8
/j_spring_security_...	3000	448	202	1159	4	7344	0.00%	13.3/sec	47.7
TOTAL	42000	633	313	1653	2	11736	0.00%	175.7/sec	902.8



With the following picture is performance analysis in which we can see that CPU are working more than 97-100% all the time. So, if we improve the resources of the CPU and improve the code of the method we said before, we can increase the concurrent users.



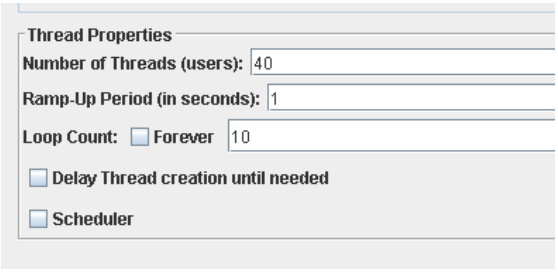
Conclusion: The maximum number of concurrent users supported by the system is 300 because the current CPU is a bottleneck.

Req. 46.1, 46.2 – An actor who is authenticated as a nutritionist must be able to Manage his or her articles, which includes listing them, showing them, creating, updating and deleting them. An article can be saved in draft mode; once they are saved in final mode, they can't be edited. Write comments about his or her articles

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

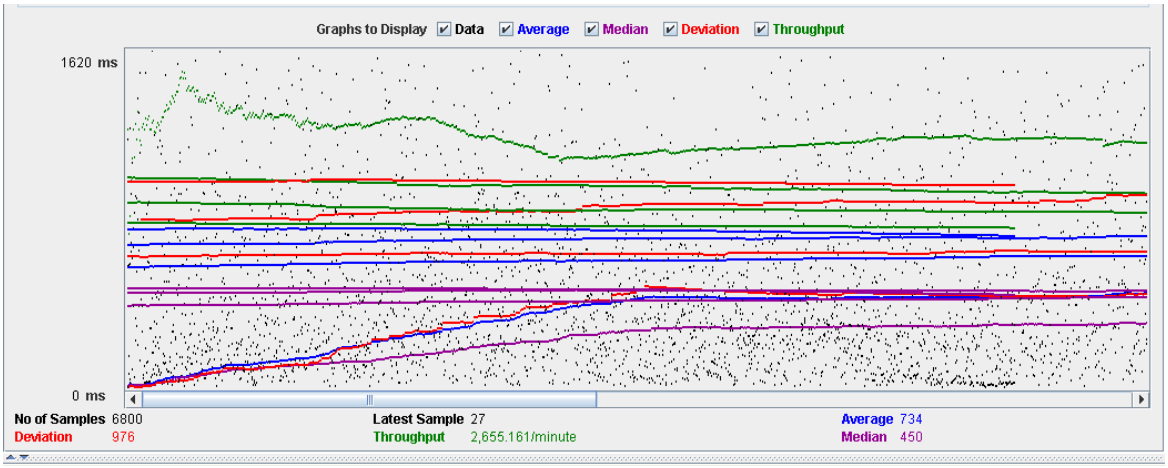
Test case description: The user nutritionist starts session, then lists the articles, creates a new article, marks it in final mode and then shows it. List the comments of the article shown and create a new comment. Finally show the new comment and close session.

Maximum workload test case: 40 concurrent users, 10 of loop count and 1 of ramp-up period.

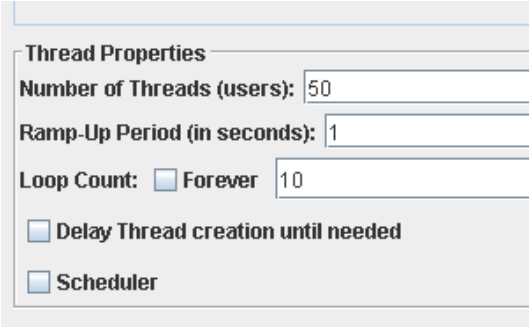


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	400	472	243	1122	10	5258	0.00%	2.6/sec	9.4
/j_spring_securit...	400	941	599	2100	17	10914	0.00%	2.6/sec	10.0
/	800	426	211	972	8	6549	0.00%	5.2/sec	18.1
/article/nutritioni...	1200	582	362	1168	16	8943	0.00%	7.8/sec	47.4
/article/nutritioni...	400	468	297	1032	13	5694	0.00%	2.6/sec	12.1
/article/nutritioni...	400	1155	809	2317	65	14839	0.00%	2.6/sec	17.3
/article/nutritioni...	400	952	610	2054	20	10746	0.00%	2.6/sec	9.7
/article/customer...	400	1004	611	2120	19	10896	0.00%	2.6/sec	9.8
/comment/custo...	800	874	585	1756	17	10590	0.00%	5.2/sec	19.6
/comment/custo...	400	771	530	1583	19	7380	0.00%	2.6/sec	9.9
/comment/custo...	400	957	678	2097	20	10421	0.00%	2.6/sec	9.9
/comment/custo...	400	965	628	1880	19	11469	0.00%	2.6/sec	9.9
/j_spring_securit...	400	447	242	969	12	6738	0.00%	2.7/sec	9.4
TOTAL	6800	734	450	1593	8	14839	0.00%	44.3/sec	191.1

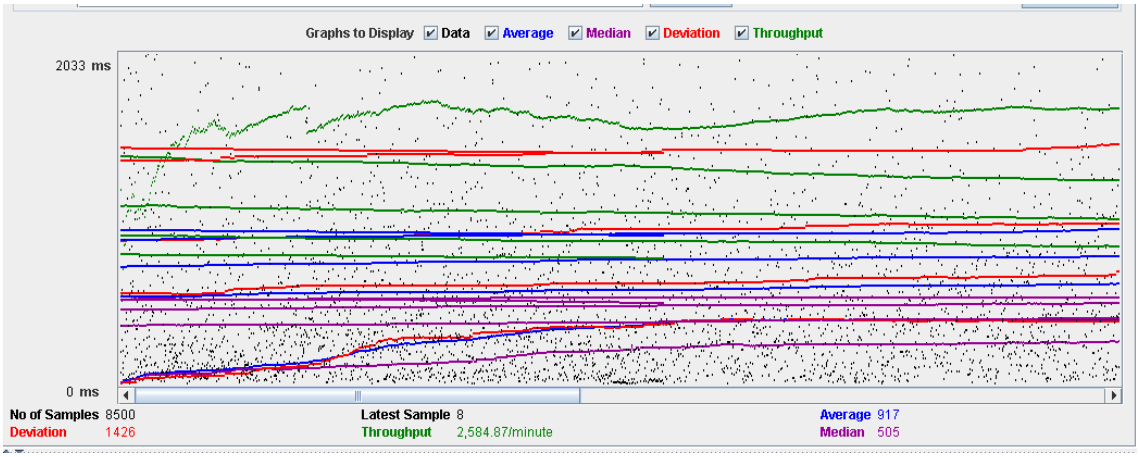


Overload test case: 50 concurrent users, 10 of loop count and 1 as ramp-up period.



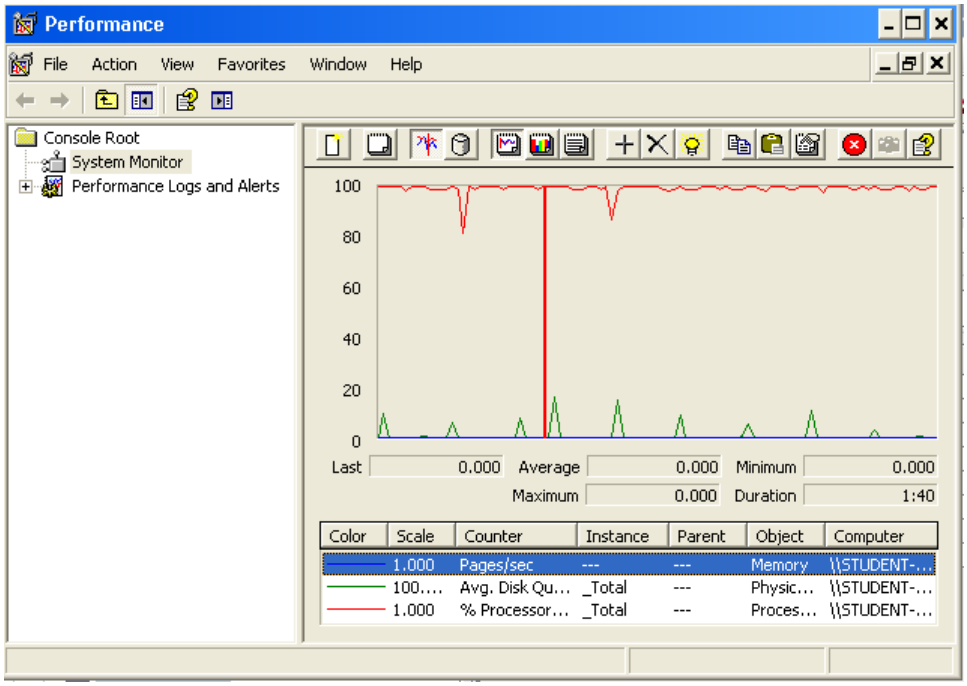
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	500	520	275	1253	9	7437	0.00%	2.6/sec	9.1
/j_spring_securit...	500	1044	629	2180	26	11143	0.00%	2.6/sec	9.7
/	1000	567	249	1270	8	9497	0.00%	5.1/sec	17.6
/article/nutritioni...	1500	719	420	1389	28	19126	0.00%	7.6/sec	46.1
/article/nutritioni...	500	610	296	1260	14	24658	0.00%	2.6/sec	11.8
/article/nutritioni...	500	1408	999	2913	95	9225	0.00%	2.6/sec	16.9
/article/nutritioni...	500	1186	674	2692	17	14804	0.00%	2.6/sec	9.5
/article/customer...	500	1178	633	2477	20	18937	0.00%	2.6/sec	9.6
/comment/custo...	1000	1091	639	2371	19	18881	0.00%	5.1/sec	19.2
/comment/custo...	500	1088	691	2314	19	16739	0.00%	2.6/sec	9.6
/comment/custo...	500	1218	734	2711	16	12128	0.00%	2.6/sec	9.6
/comment/custo...	500	1213	711	2435	18	29361	0.00%	2.6/sec	9.7
/j_spring_securit...	500	657	291	1447	13	14437	0.00%	2.6/sec	9.2
TOTAL	8500	917	505	2042	8	29361	0.00%	43.1/sec	186.1



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.

In this analysis we can also see that hard disk make some peak. We think that is when the system goes to memory to get the list.



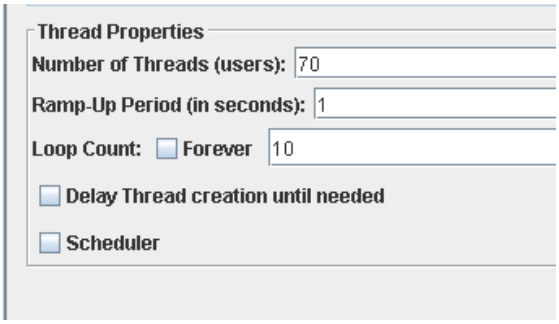
Conclusion: The maximum number of concurrent users supported by the system is 40 because the current CPU is a bottleneck.

Req. 47.1, 47.2 – An actor who is authenticated as a customer must be able to only premium customer can list and displaying the articles that are published by the nutritionist. Only premium customer can write comments in the articles.

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

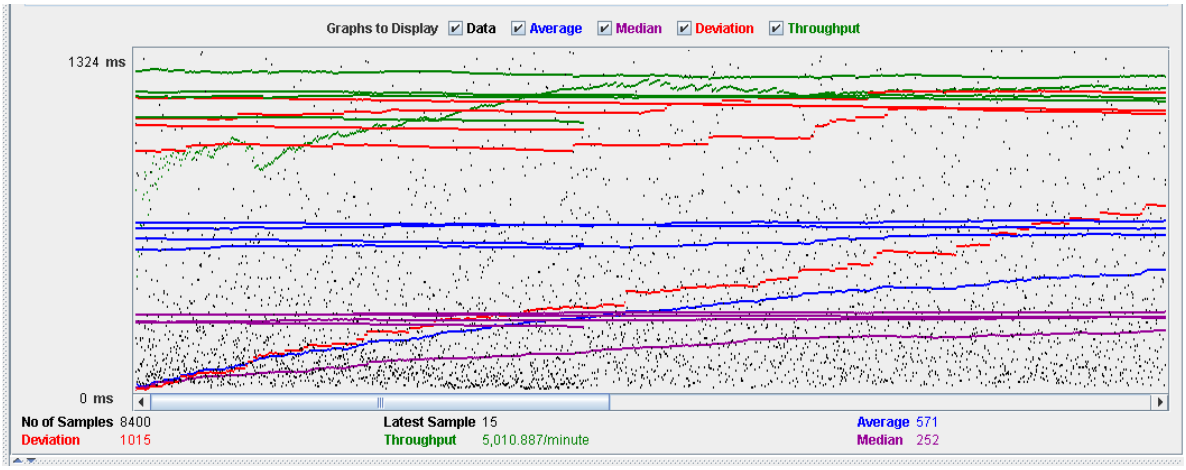
Test case description: The premium customer user starts the session, then lists the articles and shows an article. List the comments of the article shown and create a new comment. Finally show the new comment and close session.

Maximum workload test case: 70 concurrent users, 10 of loop count and 1 of ramp-up period.

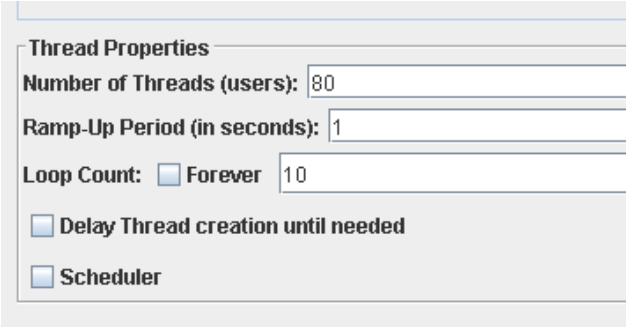


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	700	432	167	1063	9	11931	0.00%	7.0/sec	24.9
/j_spring_securit...	700	887	459	1780	15	12457	0.00%	7.0/sec	31.7
/	1400	437	168	993	9	9399	0.00%	13.9/sec	53.4
/article/customer...	700	510	214	1085	18	12369	0.00%	7.0/sec	38.4
/article/customer...	700	504	202	1183	14	8253	0.00%	7.0/sec	32.6
/comment/custo...	1400	470	213	1018	27	12590	0.00%	13.9/sec	93.1
/comment/custo...	700	462	190	1059	14	18285	0.00%	7.0/sec	34.8
/comment/custo...	700	982	564	2218	69	8609	0.00%	7.0/sec	50.6
/comment/custo...	700	852	515	1852	17	12829	0.00%	7.0/sec	31.4
/j_spring_securit...	700	411	162	1009	8	7738	0.00%	7.0/sec	25.0
TOTAL	8400	571	252	1291	8	18285	0.00%	83.5/sec	413.8

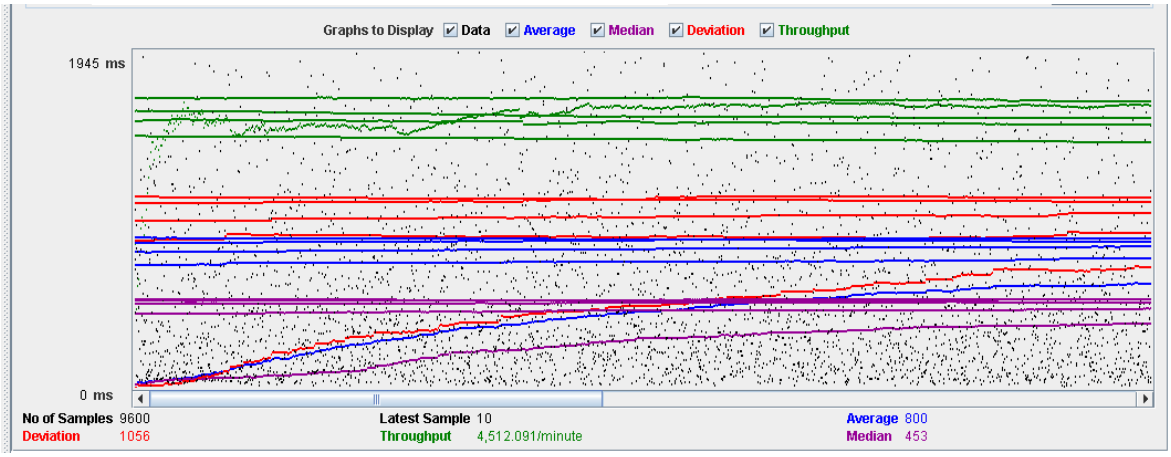


Overload test case: 80 concurrent users, 10 of loop count and 1 as ramp-up period.



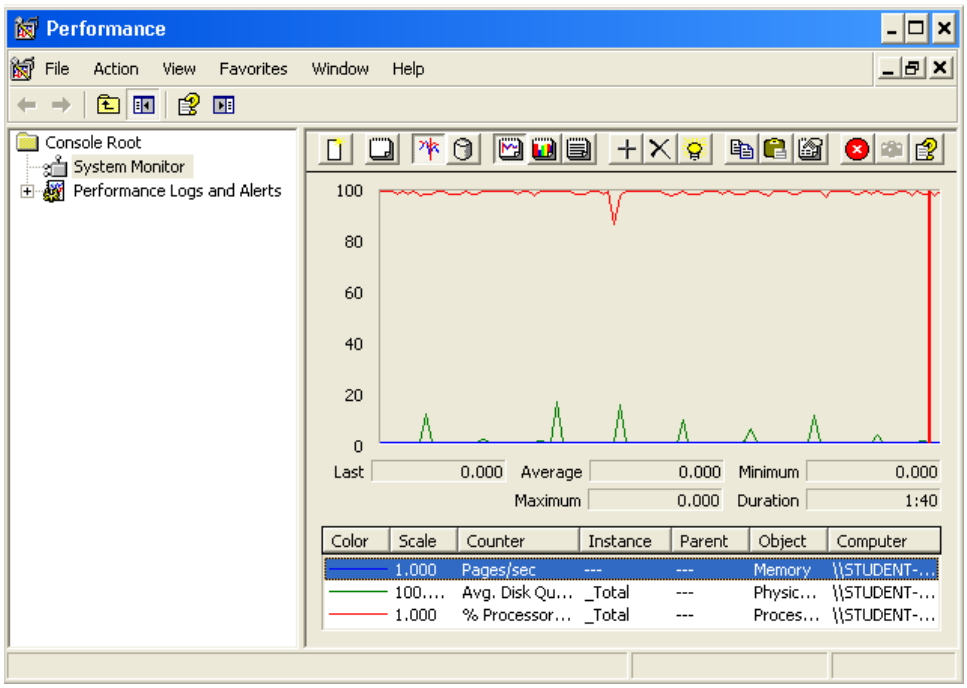
Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	800	630	295	1529	9	11562	0.00%	6.3/sec	22.4
/j_spring_securit...	800	1203	858	2618	17	13702	0.00%	6.3/sec	28.5
/	1600	612	326	1487	9	11490	0.00%	12.5/sec	48.1
/article/customer...	800	673	380	1451	19	8988	0.00%	6.3/sec	34.6
/article/customer...	800	661	350	1520	13	9978	0.00%	6.3/sec	29.3
/comment/custo...	1600	682	374	1594	22	11302	0.00%	12.6/sec	83.9
/comment/custo...	800	653	354	1568	14	6594	0.00%	6.3/sec	31.4
/comment/custo...	800	1271	879	2671	71	8334	0.00%	6.3/sec	45.5
/comment/custo...	800	1217	815	2778	16	17092	0.00%	6.3/sec	28.2
/j_spring_securit...	800	705	341	1686	12	9801	0.00%	6.3/sec	22.5
TOTAL	9600	800	453	1875	9	17092	0.00%	75.2/sec	372.7



With the following picture is performance analysis in which we can see that CPU are working more than 90% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.

In this analysis we can also see that hard disk make some peak. We think that is when the system goes to memory to get the list.



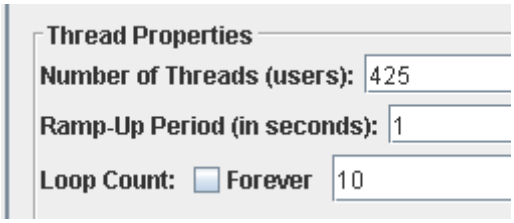
Conclusion: The maximum number of concurrent users supported by the system is 70 because the current CPU is a bottleneck.

Req. 48.1 - An actor who is authenticated as an administrator must be able to: Create an account for a new nutritionist.

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

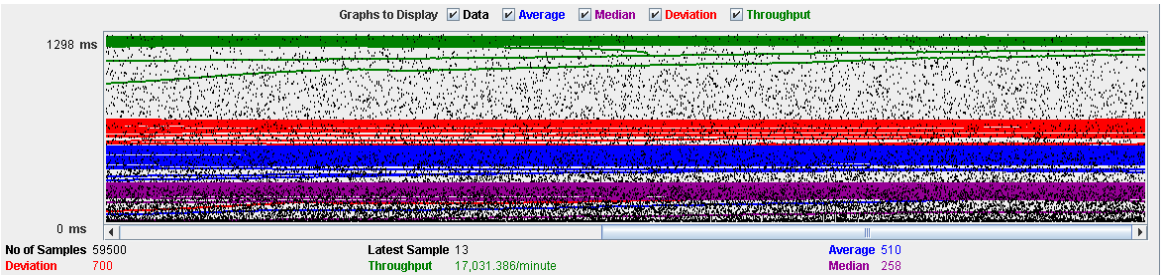
Test case description: The user logs in the system as an administrator, then, he/she clicks on *Register nutritionist* in the main menu, he/she fills the form and save it. Next, the administrator logs out the system and logs in with the new account.

Maximum workload test case: 425 concurrent users, 10 of loop count and 1 of ramp-up period.

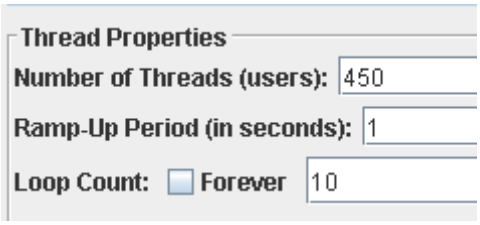


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	21250	431	208	1113	3	11243	0.00%	101.4/sec	363.(
/security/login.do	8500	421	201	1105	2	7282	0.00%	41.5/sec	148.(
/j_spring_security_...	8500	877	586	2067	7	8793	0.00%	41.5/sec	169.(
/actor/administrato...	8500	490	247	1240	5	8435	0.00%	42.7/sec	347.(
/welcome/index.do	4250	470	239	1191	4	7965	0.00%	21.6/sec	89.(
/j_spring_security_...	8500	472	244	1205	4	11656	0.00%	42.1/sec	151.(
TOTAL	59500	510	258	1310	2	11656	0.00%	283.9/sec	1233.(

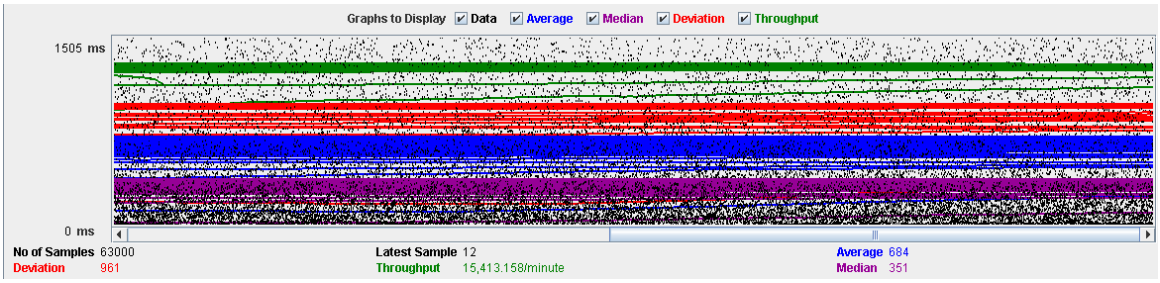


Overload test case: 450 concurrent users, 10 of loop count and 1 as ramp-up period.

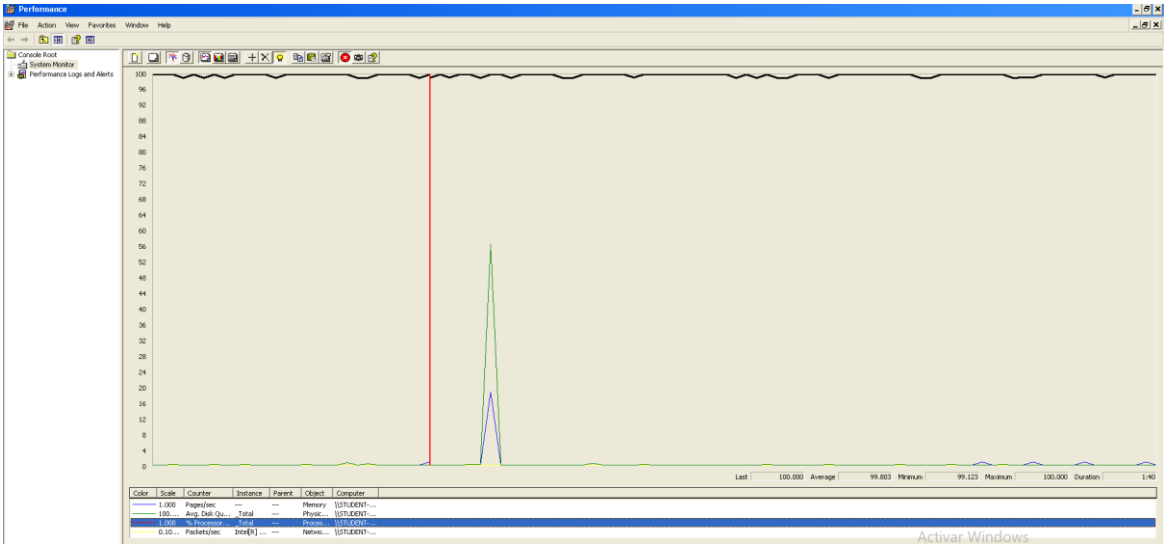


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "j_spring_security_check". However, this method is an internal unit of Spring. That's means that we cannot improve the performance by refactoring the code.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	22500	583	289	1445	2	13629	0.00%	91.7/sec	328.6
/security/login.do	9000	573	282	1456	2	12238	0.00%	37.4/sec	133.4
/j_spring_security_...	9000	1180	796	2658	5	15192	0.00%	37.4/sec	153.1
/actor/administrato...	9000	631	320	1552	5	11685	0.00%	38.4/sec	312.2
/welcome/index.do	4500	604	304	1540	3	9855	0.00%	19.3/sec	79.8
/j_spring_security_...	9000	647	338	1576	4	15195	0.00%	37.9/sec	136.0
TOTAL	63000	684	351	1703	2	15195	0.00%	256.9/sec	1116.2



With the following picture is performance analysis in which we can see that CPU are working almost 100% all the time. So, if we improve the resources of the CPU are possible that the system supports more user concurrently.



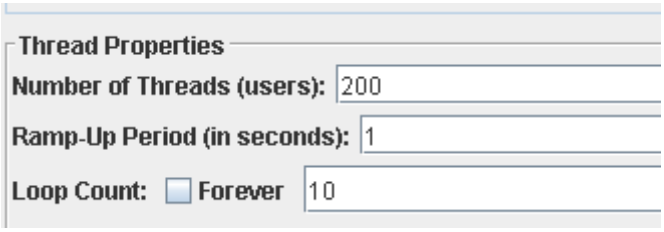
Conclusion: The maximum number of concurrent users supported by the system is 425 because the current CPU is a bottleneck.

Req. 48.2 - An actor who is authenticated as an administrator must be able to: Launch a process that computes an internal score for every trainer.

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

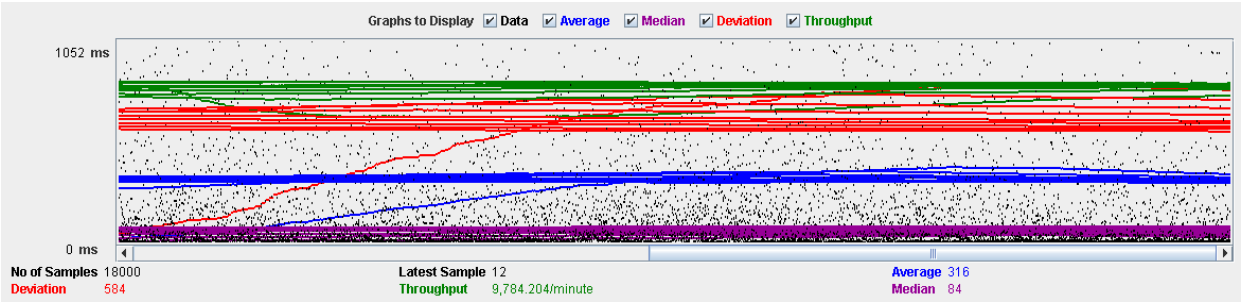
Test case description: The user logs in the system as an administrator, then, he/she clicks on *Actor list* in the main menu and launch score process.

Maximum workload test case: 200 concurrent users, 10 of loop count and 1 of ramp-up period.

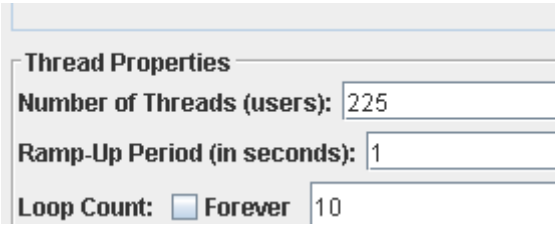


With this configuration we make sure that the system doesn't produce any errors and has a good response time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	6000	194	43	545	3	4891	0.00%	54.4/sec	197.3
/security/login.do	2000	127	12	324	2	3652	0.00%	19.7/sec	70.2
/j_spring_security_...	2000	267	65	742	5	4774	0.00%	19.5/sec	84.4
/actor/administrato...	4000	342	127	909	12	5943	0.00%	38.2/sec	266.1
/actor/administrato...	2000	923	537	2344	26	5934	0.00%	19.4/sec	144.5
/j_spring_security_...	2000	259	74	731	4	4095	0.00%	19.4/sec	69.3
TOTAL	18000	316	84	906	2	5943	0.00%	163.1/sec	792.2

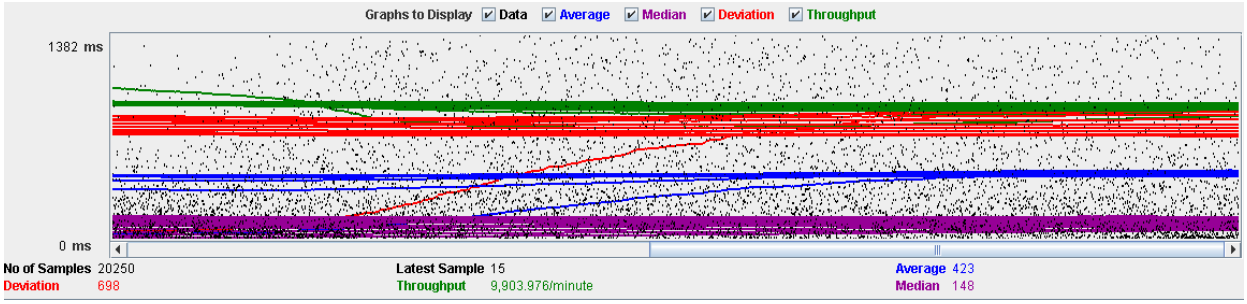


Overload test case: 225 concurrent users, 10 of loop count and 1 as ramp-up period.

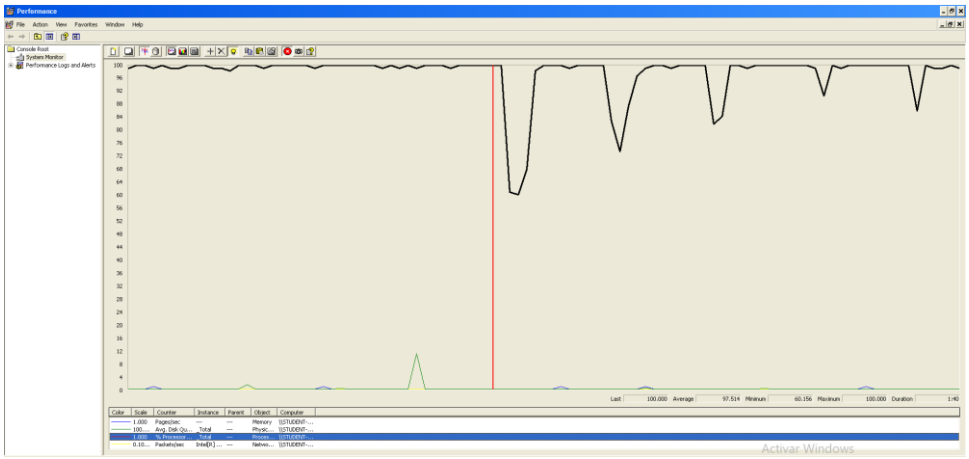


Although there aren't errors, the 90%-line time is not acceptable. The requests with highest time are "/actor/administrator/scoreProcess.do". However, this method is an implementation of the developer team so we can try to refactoring it to improve its time.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	6750	274	84	789	2	6970	0.00%	55.0/sec	199.
/security/login.do	2250	194	42	537	2	3430	0.00%	19.5/sec	69.
/j_spring_security_...	2250	443	160	1195	5	7972	0.00%	19.5/sec	84.
/actor/administrato...	4500	439	190	1164	11	5654	0.00%	38.5/sec	268.
/actor/administrato...	2250	1099	689	2718	28	6560	0.00%	19.5/sec	145.
/j_spring_security_...	2250	370	145	1007	4	5046	0.00%	19.5/sec	69.
TOTAL	20250	423	148	1190	2	7972	0.00%	165.1/sec	802.



With the following picture is performance analysis in which we can see that CPU is not working at 100%. So, if we improve improve our code sure we will obtained more concurrent users for this test.



Conclusion: The maximum number of concurrent users supported by the system is 200 because the time is accepted.