Non Functional Testing

for

Academic Affairs System

Introduction

Non-Functional Testing is defined as a type of Software testing to check non-functional aspects of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

In order to test our system for non functional requirements we performed a series of different tests, all of which tested different aspects of non functional requirements.

We used Jmeter in order to perform a variety of tests

All the following tests were conducted on a deployed render website.

Some of the other tests were done and validated manually

1. Performance Testing

The performance tests help to identify the software design and architecture performance issue.

In order to check for the performance of the website, we manually checked for the decided parameters and compared the outputs with our desired results.

Following were the parameters on the basis of which performance was measured along with their outputs:-

1. Response time: For our website, the desired response time was less than 5 seconds for all the functionality. The testing was done for a single user in order to validate if the website works properly and within the set bounds. The output of the test was that all the features work with a response time of less than 3 seconds.

2. Identifying Failure Points and bottlenecks: For the website, the major failure points were when the load on the server is high. As the hosting is done on a free server which is not a dedicated server for this website, when the load on the server is very high the server starts to fail for https get requests and bottleneck in performance is caused too.

2. Load Testing

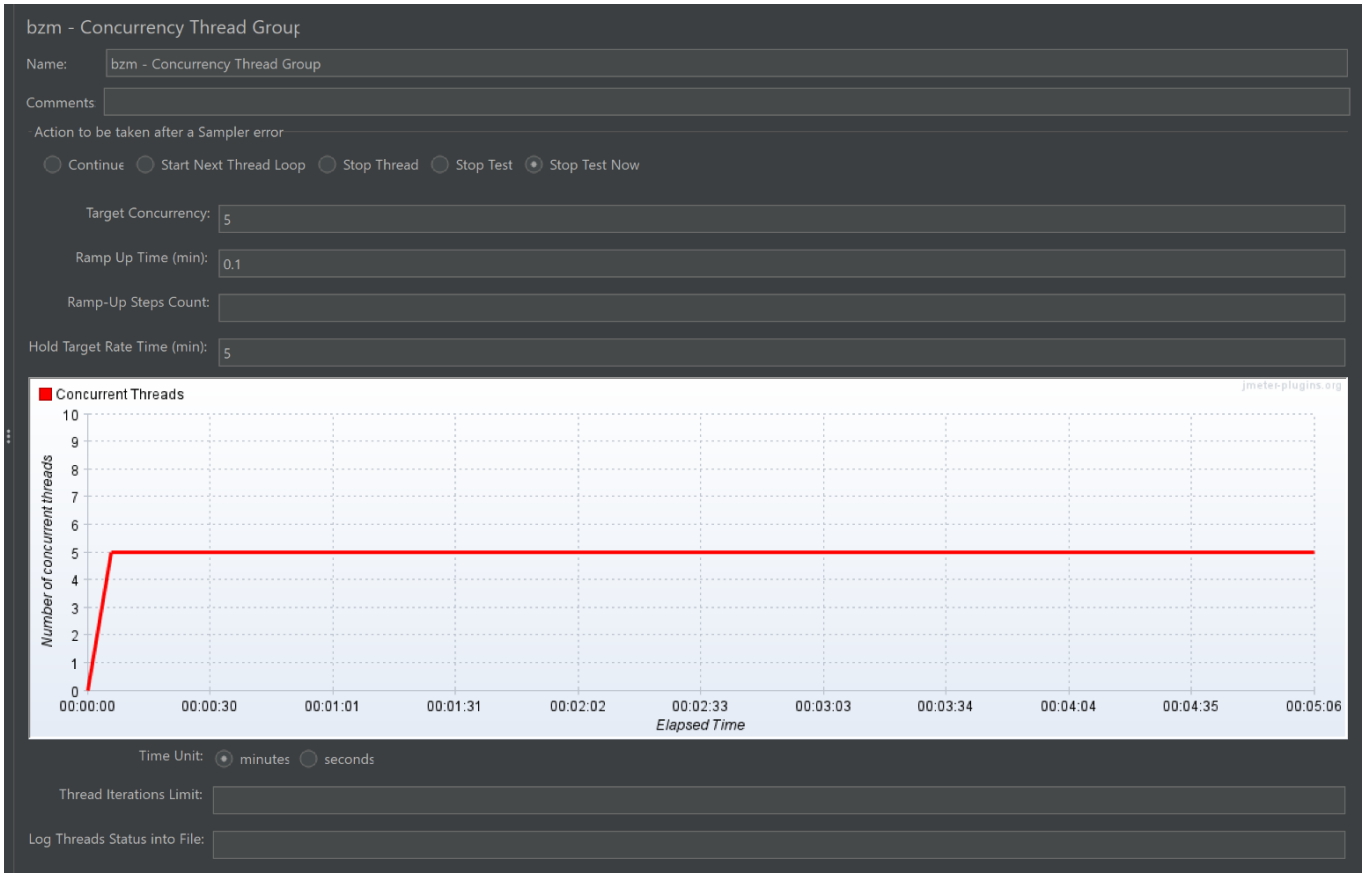
Load testing checks how the software behaves under both normal and peak conditions. In order to do the load test, we used the jmeter application which helps to run different threads for performing test cases. Concurrency threads were used to simulate the concurrent users using the website.

Our Desired Response time is less than 5 seconds

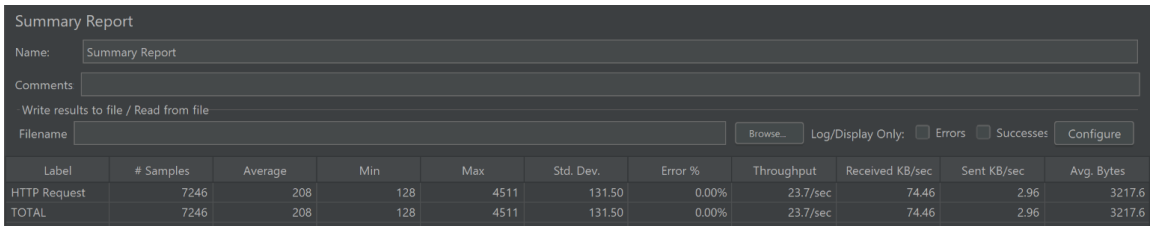
Note: While running the threads, the website was also tested manually to find if the test results were inline with the manual experience.

<u>Test case 1</u>

Thread Configuration:

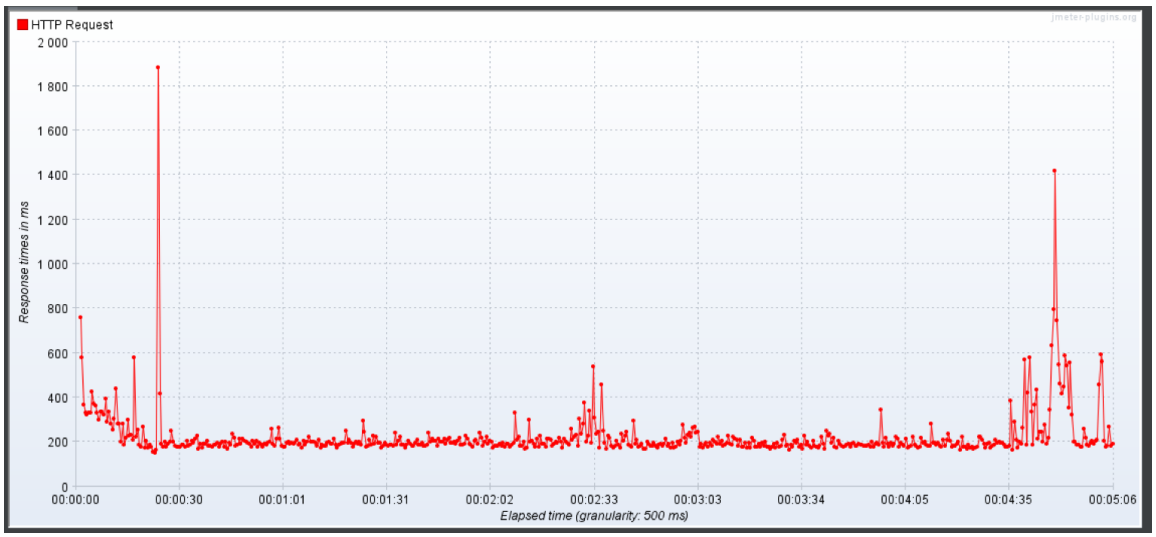


Summary Report

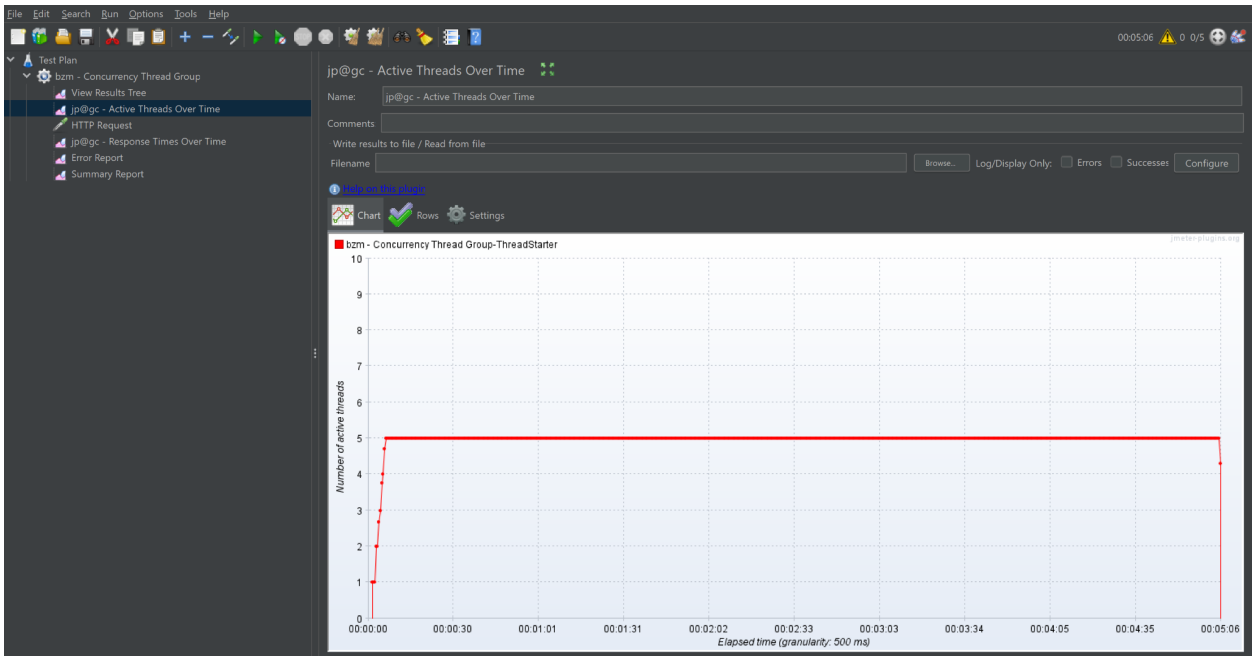


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	7246	208	128	4511	131.50	0.00%	23.7/sec	74.46	2.96	3217.6
TOTAL	7246	208	128	4511	131.50	0.00%	23.7/sec	74.46	2.96	3217.6

Response Times vs Time Graph



Active Threads vs Time Graph

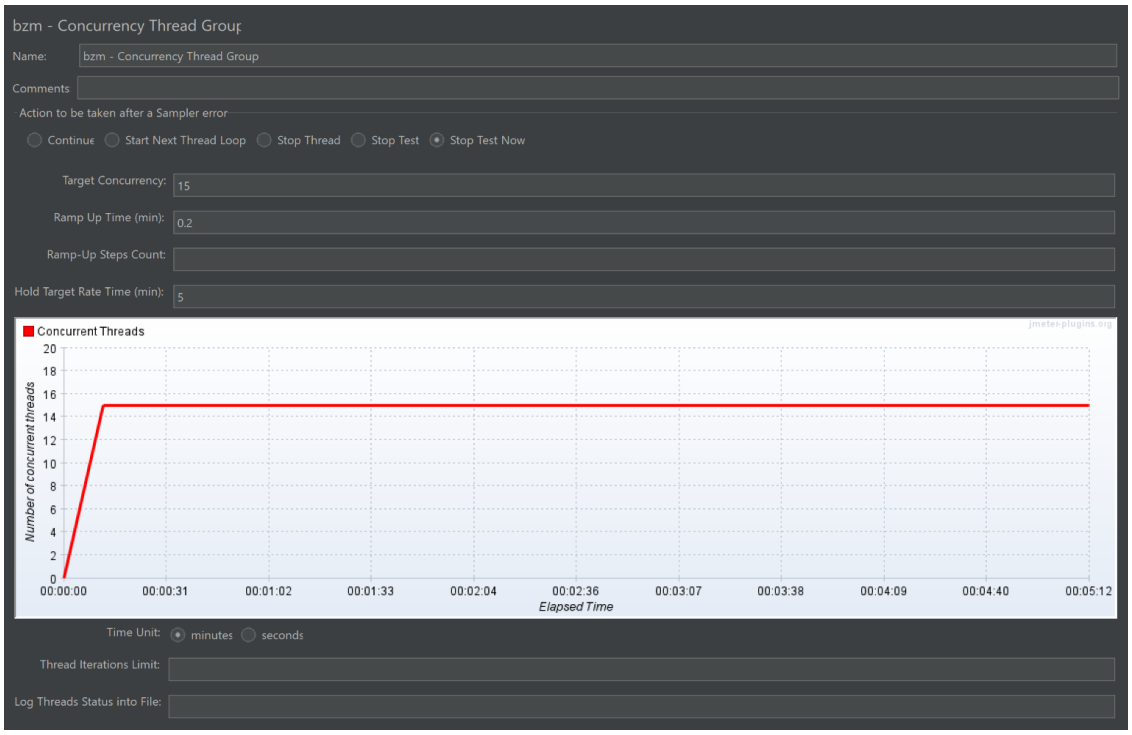


Result of Test 1: From the above graphs and reports, it is easy to observe that the application is able to handle 5 concurrent users and keep the response times within the desired thresholds. It is inline with the manual testing that was performed when these threads were running.

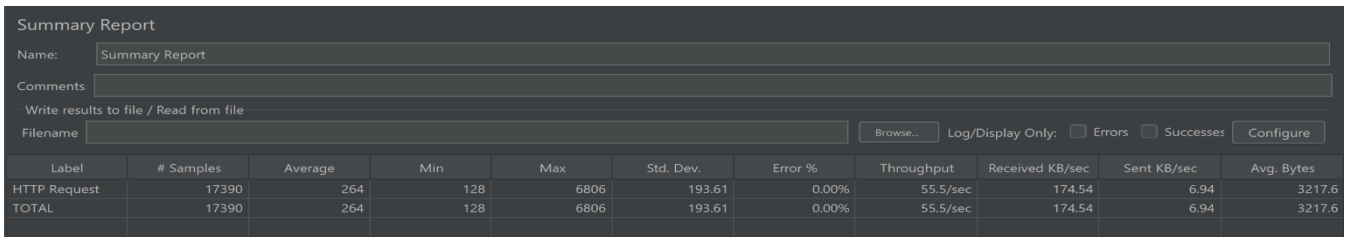
In order to find the minimum load at which our threshold is broken, we increase the number of concurrent threads in the next test case.

Test Case 2:

Thread Configuration:

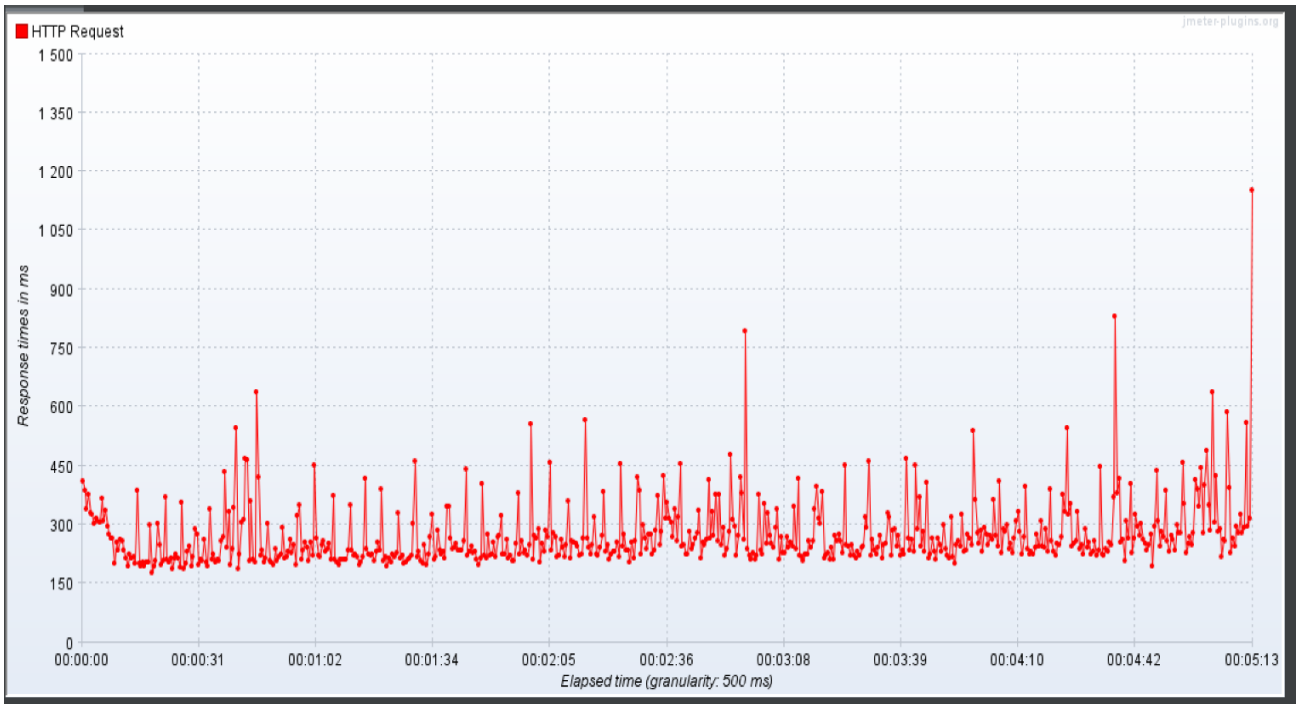


Summary Report

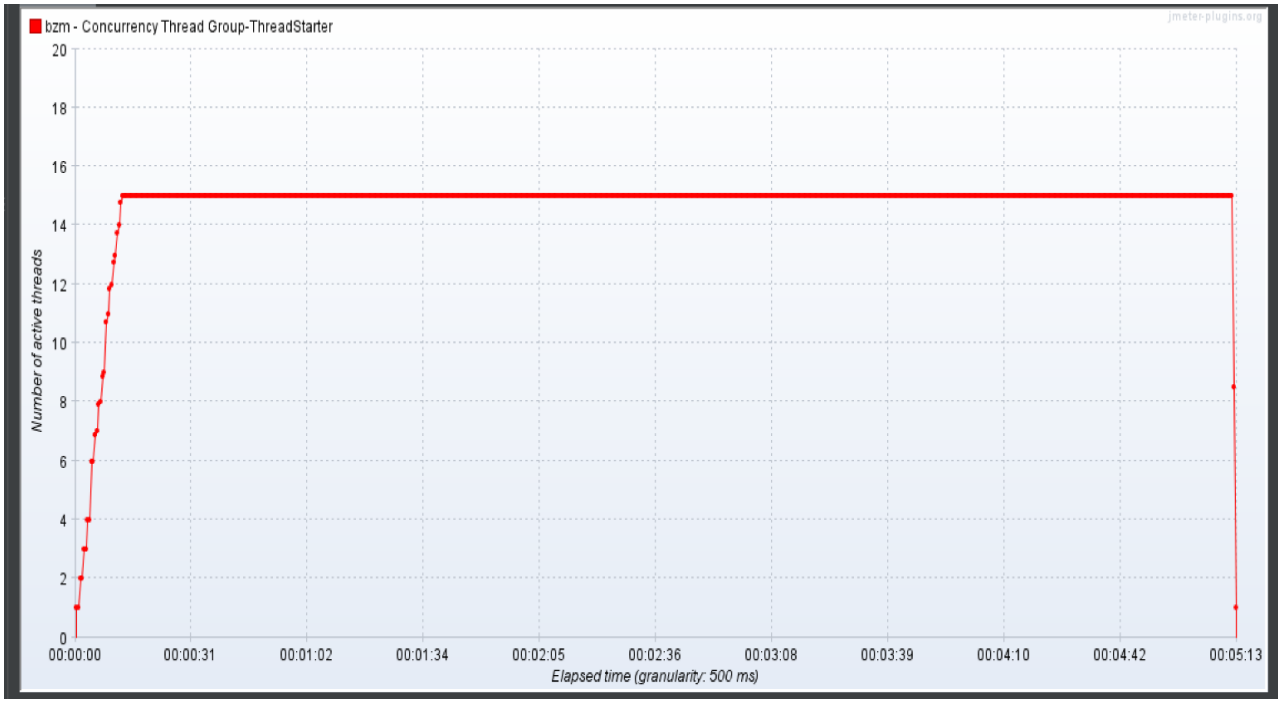


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	17390	264	128	6806	193.61	0.00%	55.5/sec	174.54	6.94	3217.6
TOTAL	17390	264	128	6806	193.61	0.00%	55.5/sec	174.54	6.94	3217.6

Response Times VS Time Graph



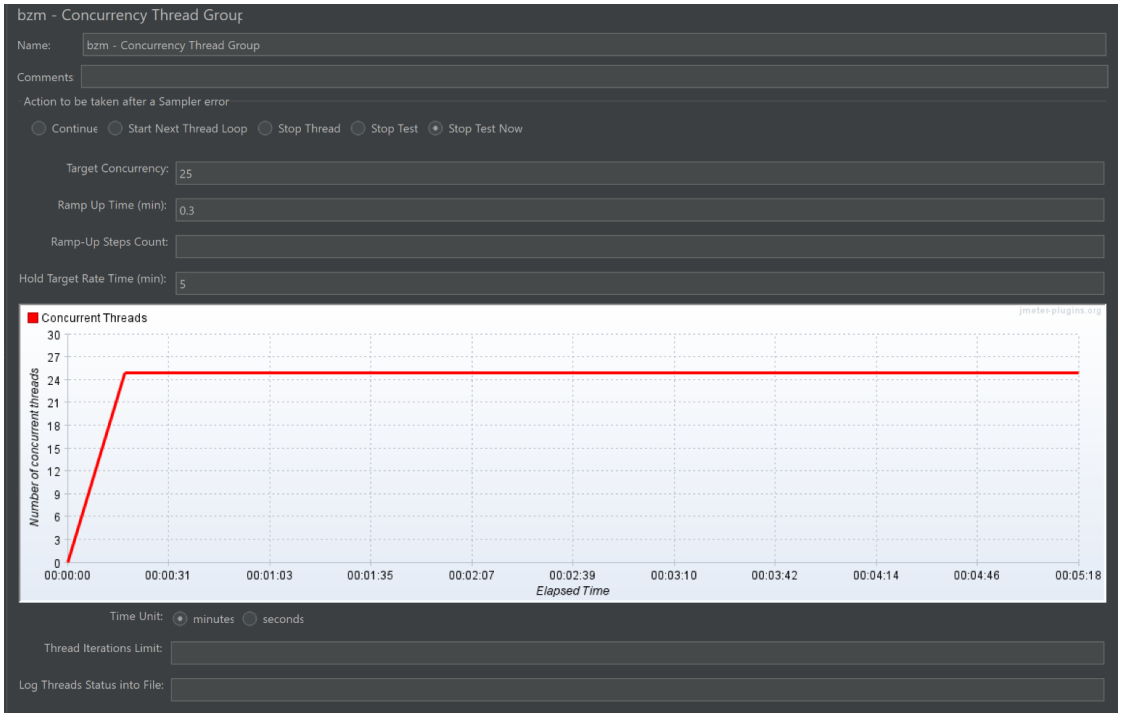
Active Threads VS Time Graph



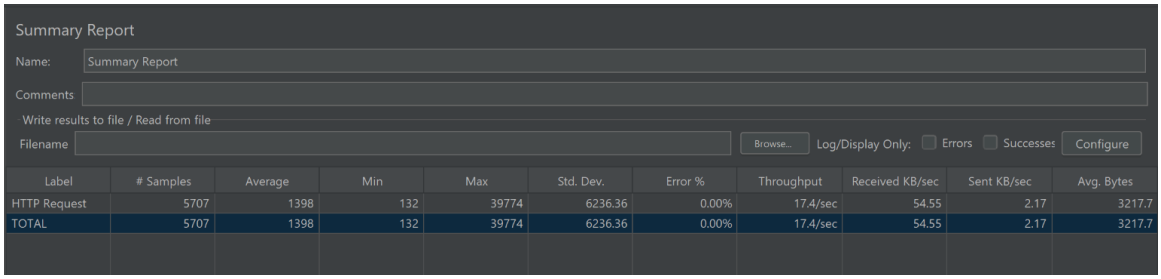
Result from test 2:
From the above graphs and reports, it is easy to observe that the application is able to handle 15 concurrent users and keep the response times within the desired thresholds. It is inline with the manual testing that was performed when these threads were running.
In order to find the minimum load at which our threshold is broken, we increase the number of concurrent threads in the next test case.

Test Case 3:

Thread Configuration:

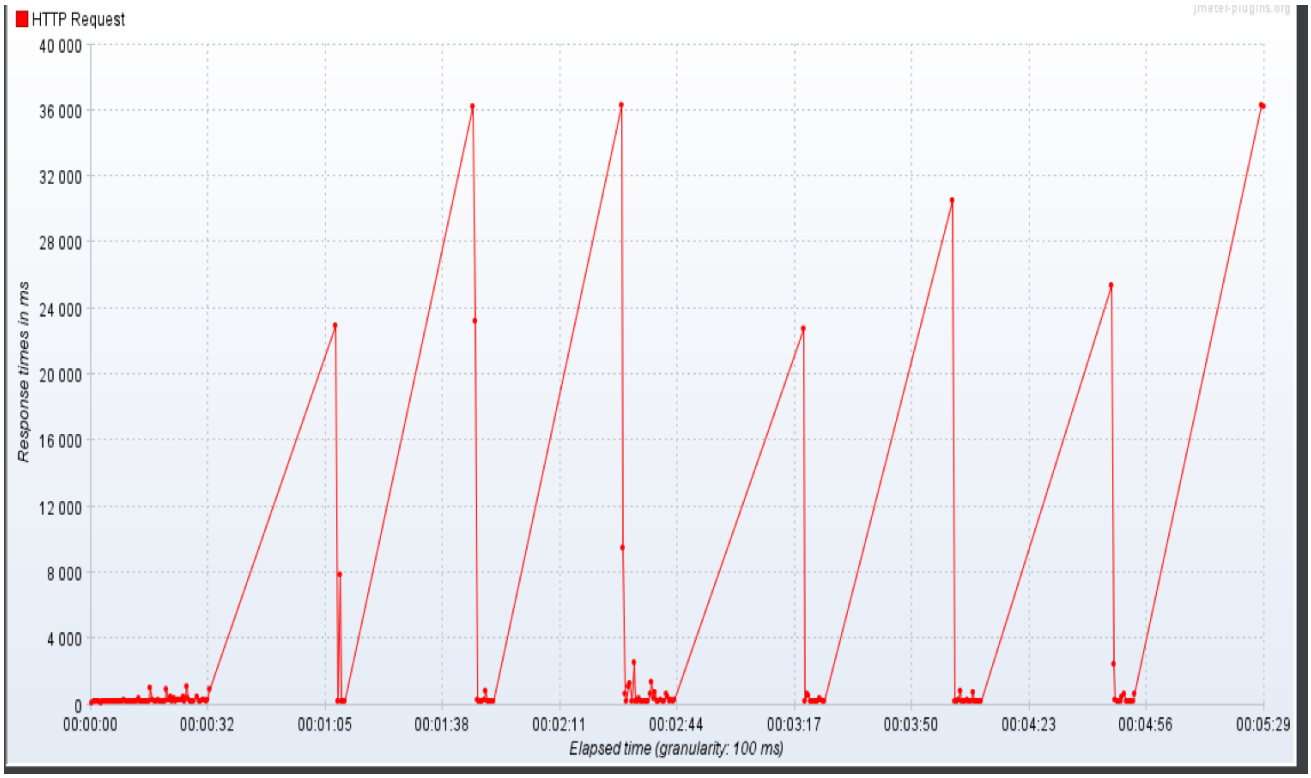


Summary Report:

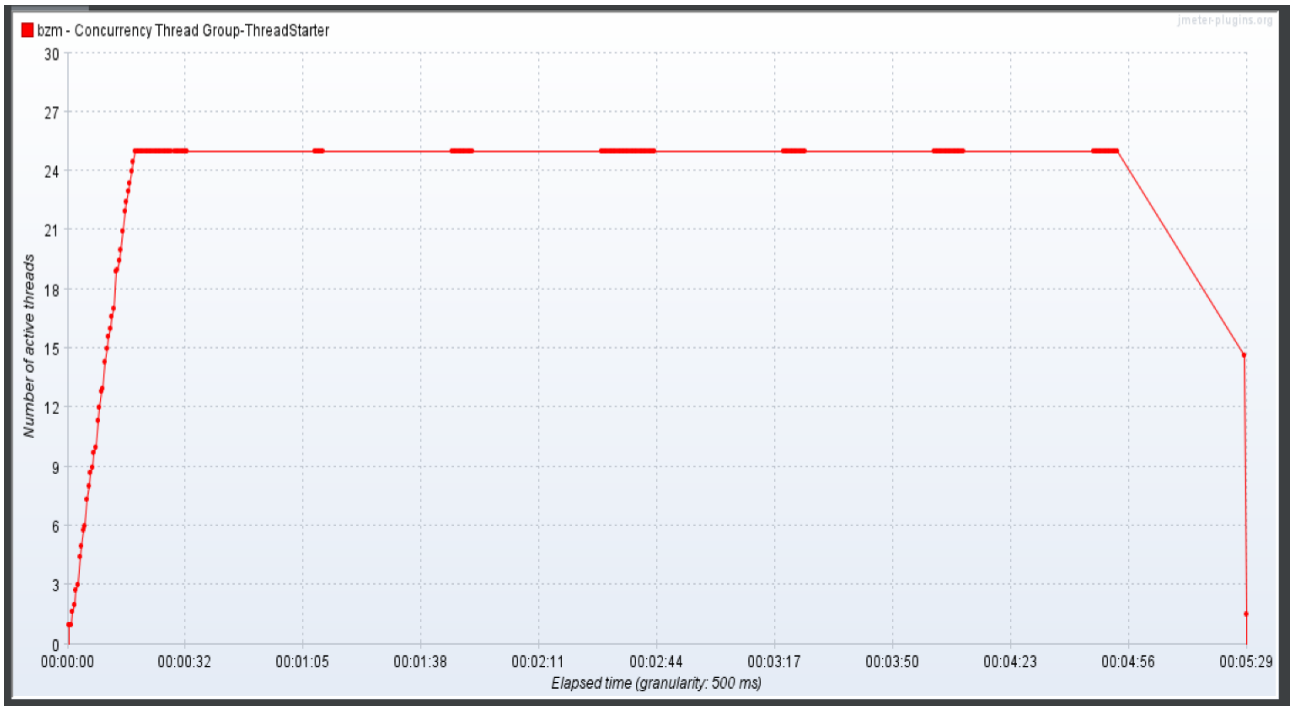


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	5707	1398	132	39774	6236.36	0.00%	17.4/sec	54.55	2.17	3217.7
TOTAL	5707	1398	132	39774	6236.36	0.00%	17.4/sec	54.55	2.17	3217.7

Response Times VS Time Graph



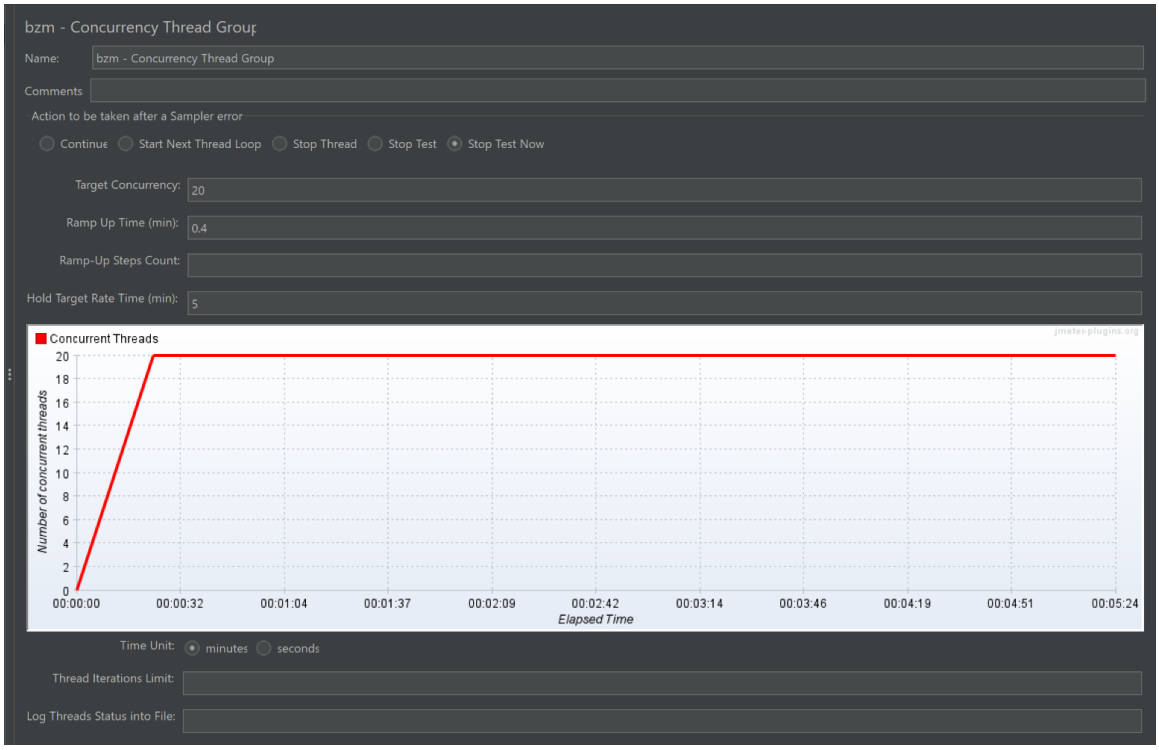
Active Threads VS Time Graph



Result from test 3:
From the above graphs and reports, we can see that a significant amount of requests are peaking the response times well above the threshold value and hence in order to calculate the optimal load value we decrease the number of concurrent threads.

Test Case 4:

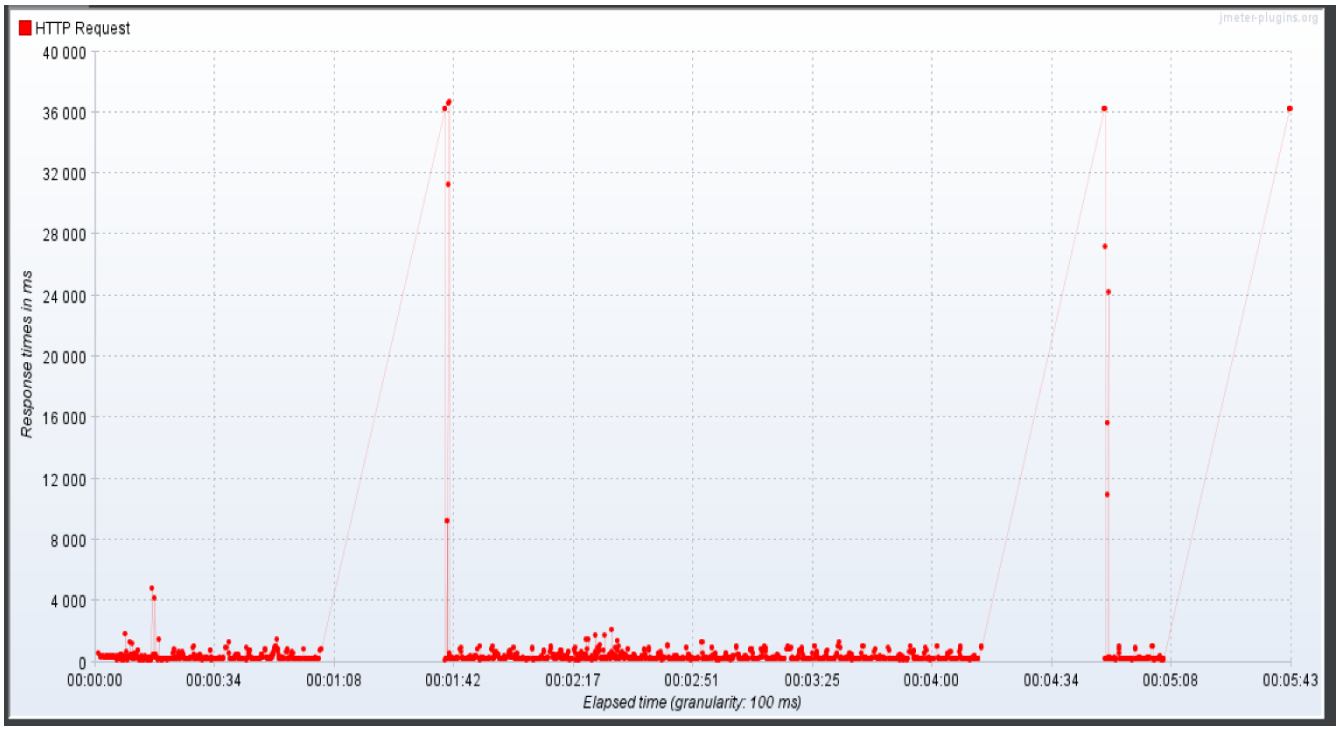
Thread Configuration:



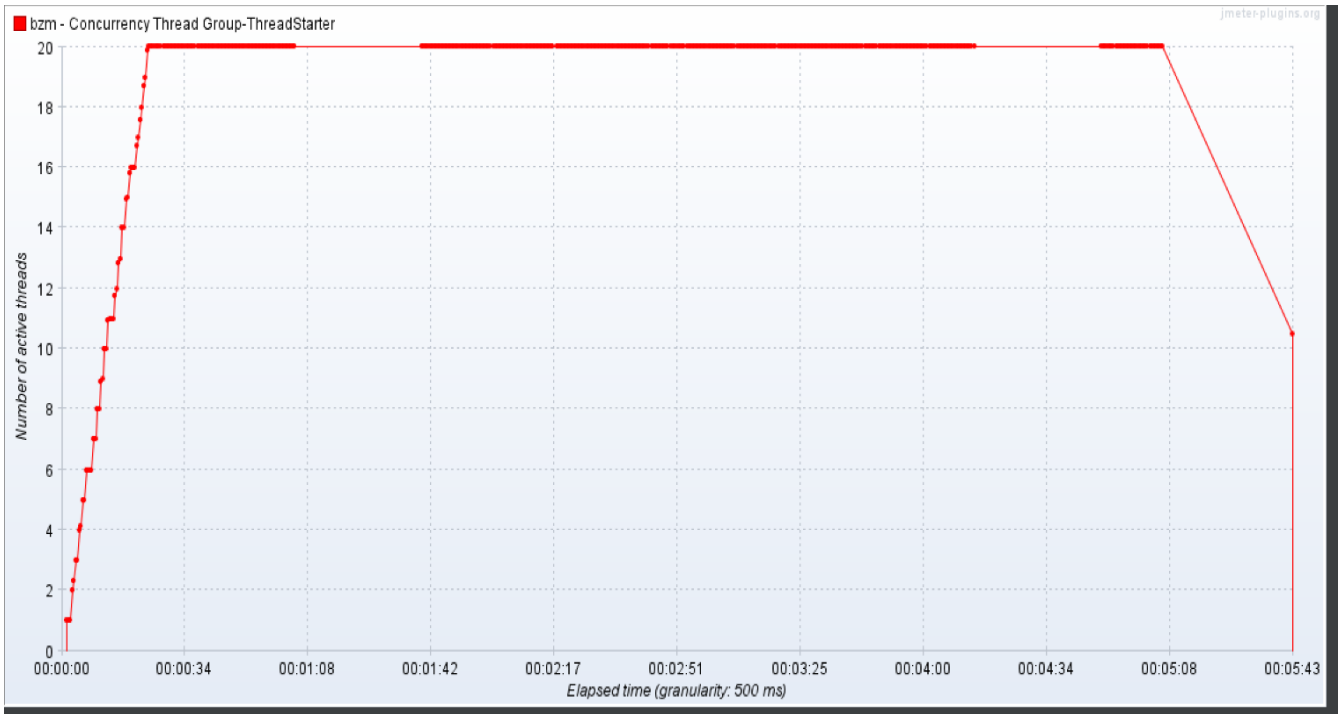
Summary Report

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	14890	443	132	36743	2296.25	0.00%	43.5/sec	136.63	5.44	3217.6
TOTAL	14890	443	132	36743	2296.25	0.00%	43.5/sec	136.63	5.44	3217.6

Response Times VS Time Graph



Active Threads VS Time Graph



Result from test 4:
From the above graphs and reports, we can see that a significant amount of requests are peaking the response times well above the threshold value.

From the above 4 test cases it can be concluded that for our desired threshold value the web application can run approximately 15-17 concurrent users.

3. Stress Testing

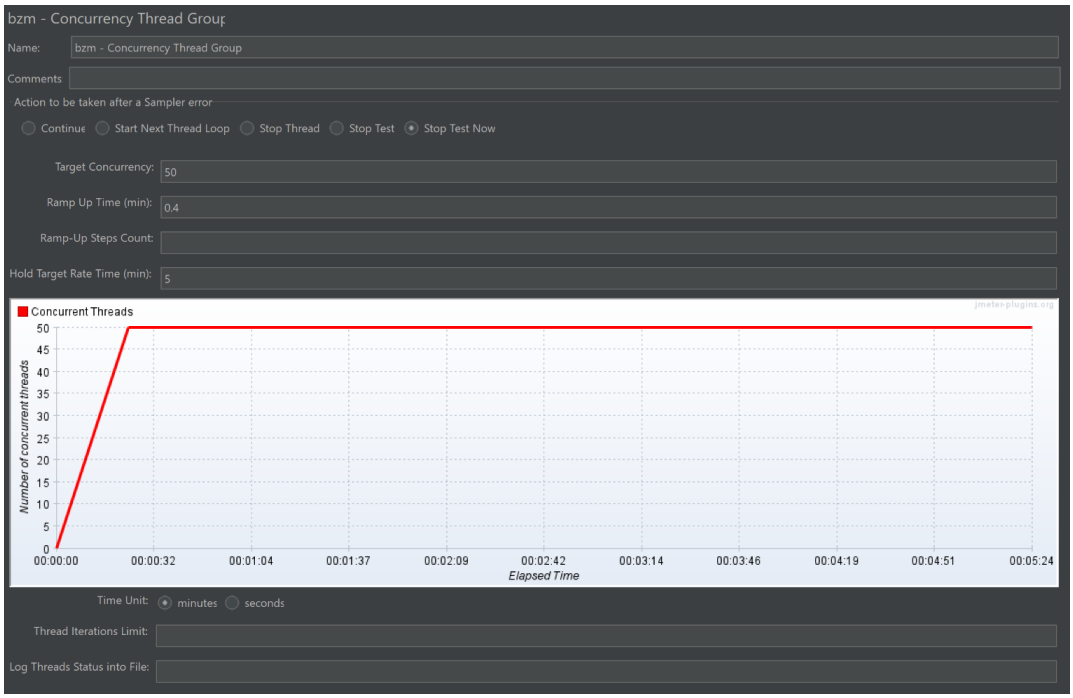
Stress testing checks how the software behaves under abnormal conditions. This determines the limit at which the software will break.

In order to do the load test, we used the jmeter application which helps to run different threads for performing test cases. Concurrency threads were used to simulate the concurrent users using the website.

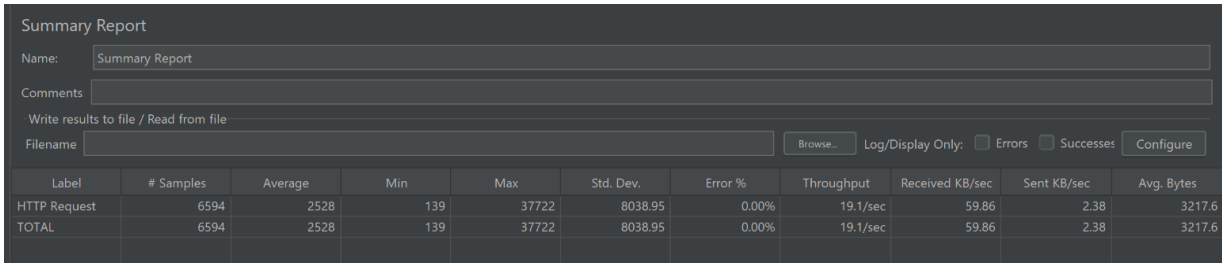
Note: While running the threads, the website was also tested manually to find if the test results were inline with the manual experience.

Test case 1

Thread Configuration:

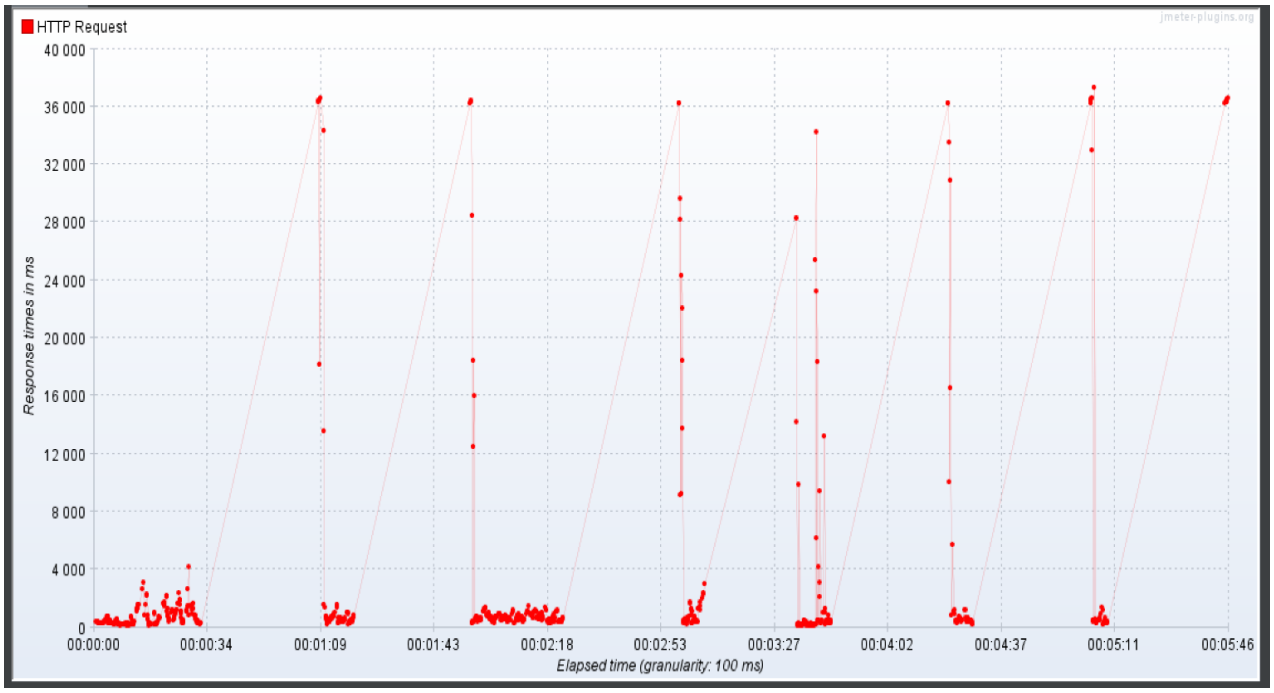


Summary Report:

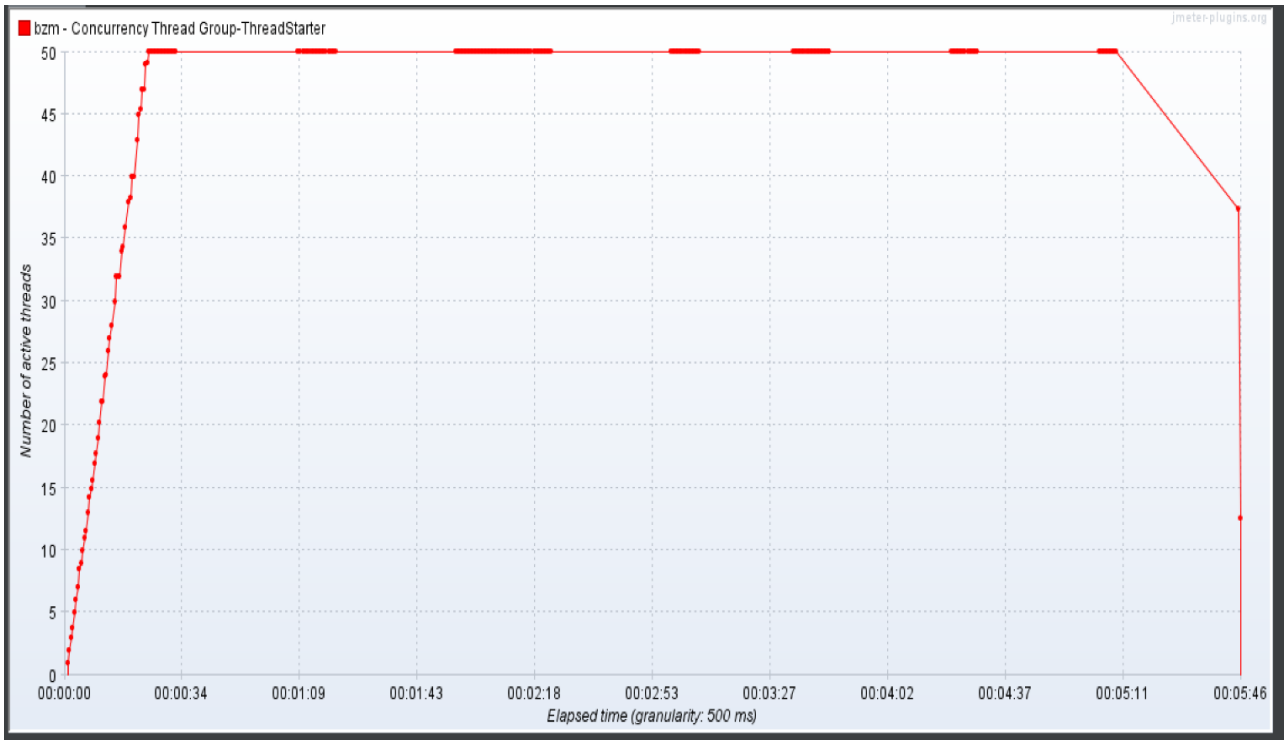


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	6594	2528	139	37722	8038.95	0.00%	19.1/sec	59.86	2.38	3217.6
TOTAL	6594	2528	139	37722	8038.95	0.00%	19.1/sec	59.86	2.38	3217.6

Response Times VS Time Graph



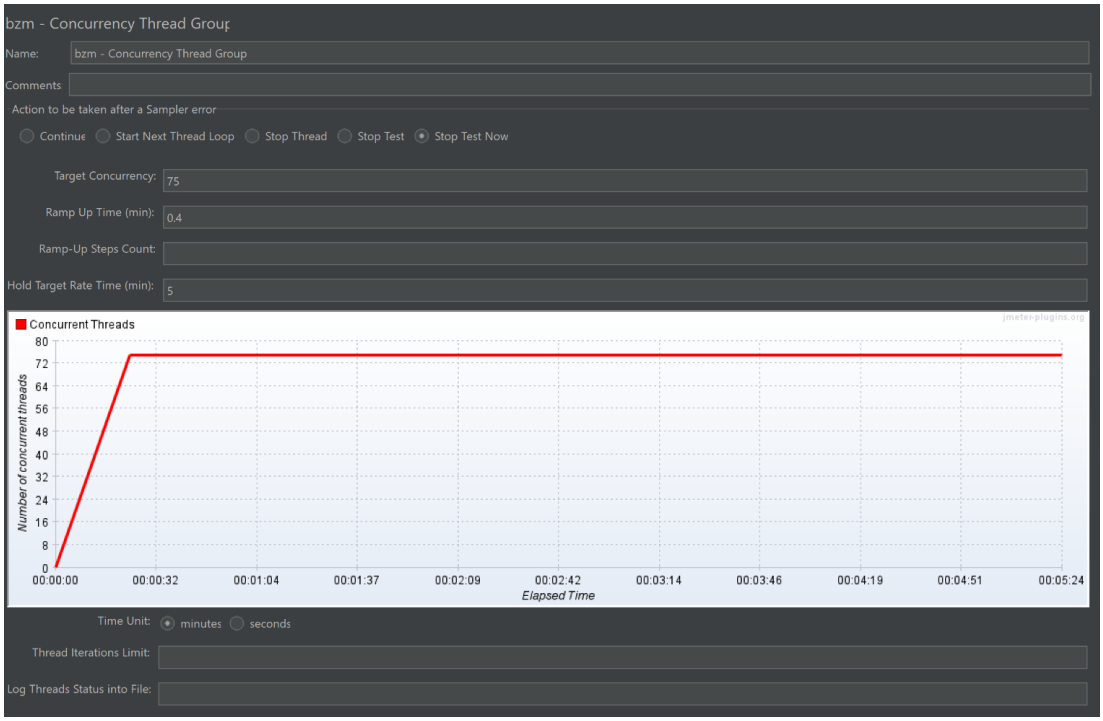
Active Threads VS Time Graph



Result from test 1:
From the above graphs and reports, we can see that the system can handle 50 concurrent users although it might not be able to keep them within the response time threshold. All the functionality were working correctly when checked with this load.
In order to get the break point we increase the number of threads in the next test case.

Test Case 2:

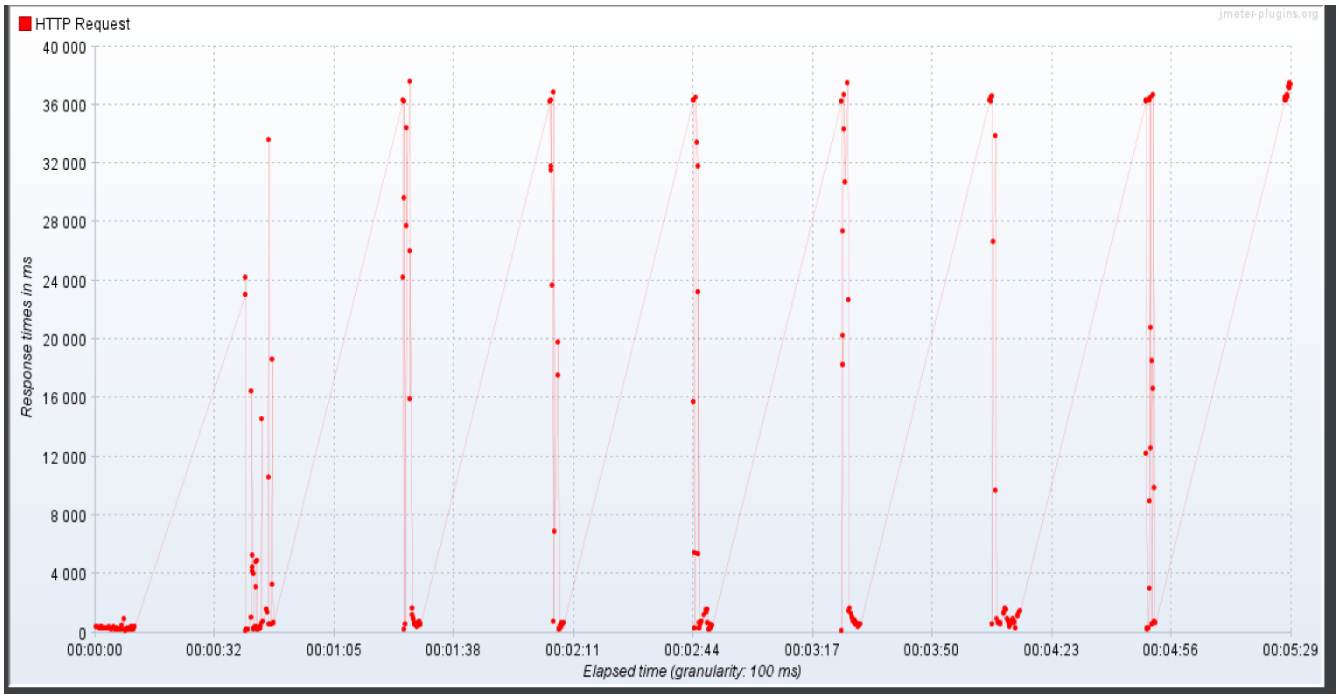
Thread Configuration:



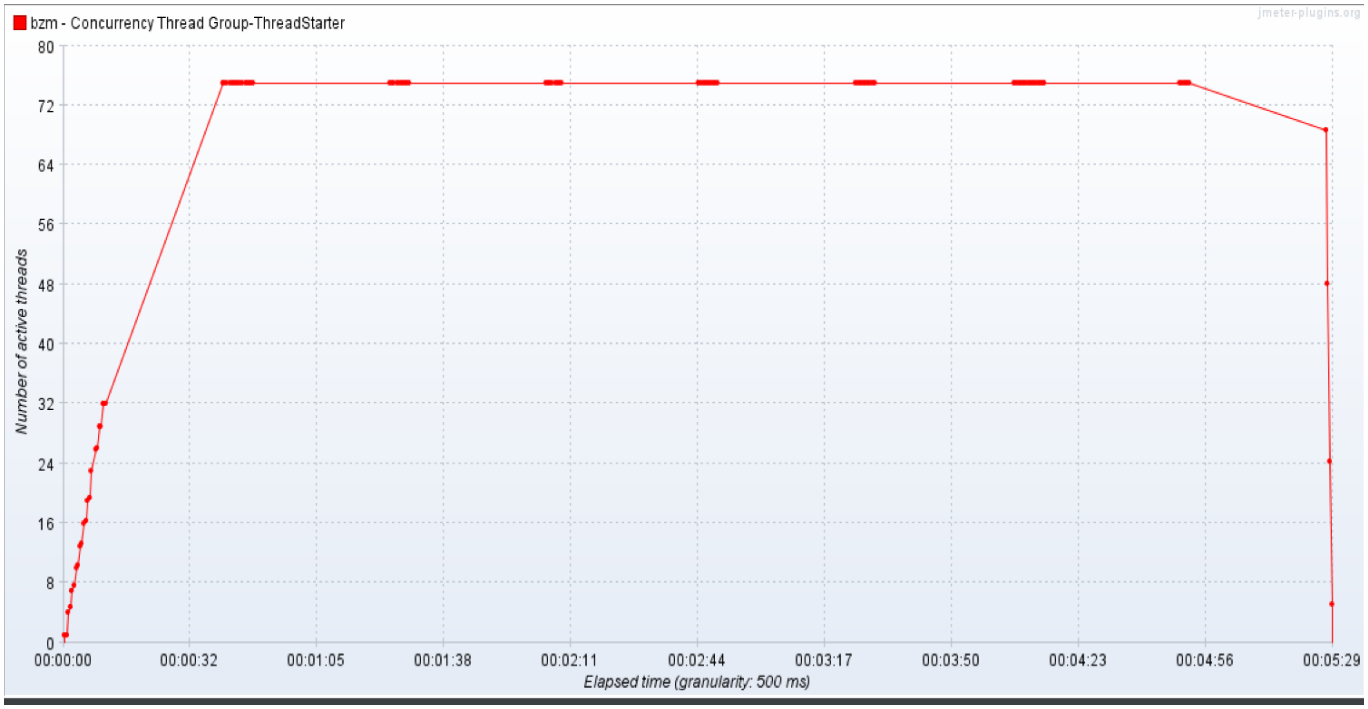
Summary Report

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	3719	6377	143	38134	13099.77	0.00%	11.3/sec	35.46	1.41	3217.7
TOTAL	3719	6377	143	38134	13099.77	0.00%	11.3/sec	35.46	1.41	3217.7

Response Times VS Time Graph



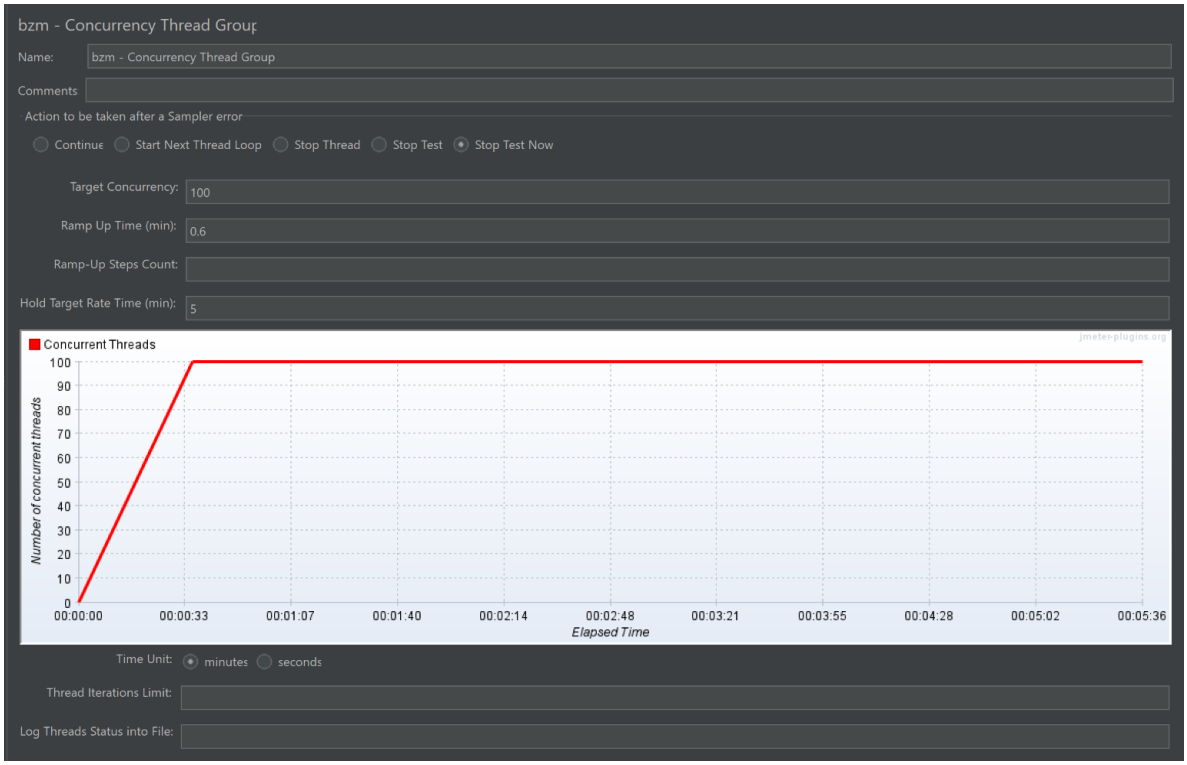
Active Threads VS Time Graph



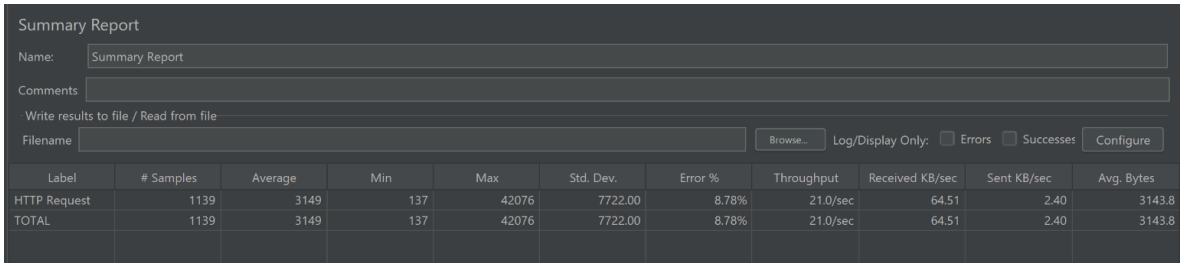
Result from test 2:
From the above graphs and reports, we can see that the system can handle 75 concurrent users although it might not be able to keep them within the response time threshold. All the functionality were working correctly when checked with this load.
In order to get the break point we increase the number of threads in the next test case.

Test Case 3:

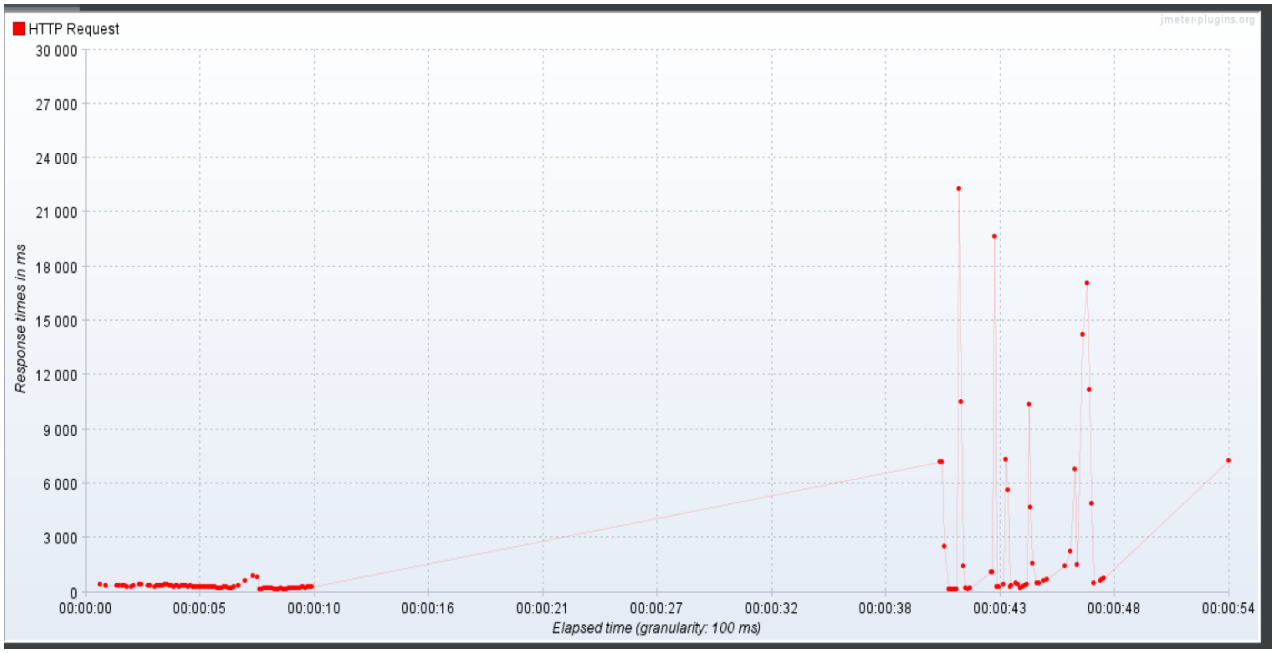
Thread Configuration:



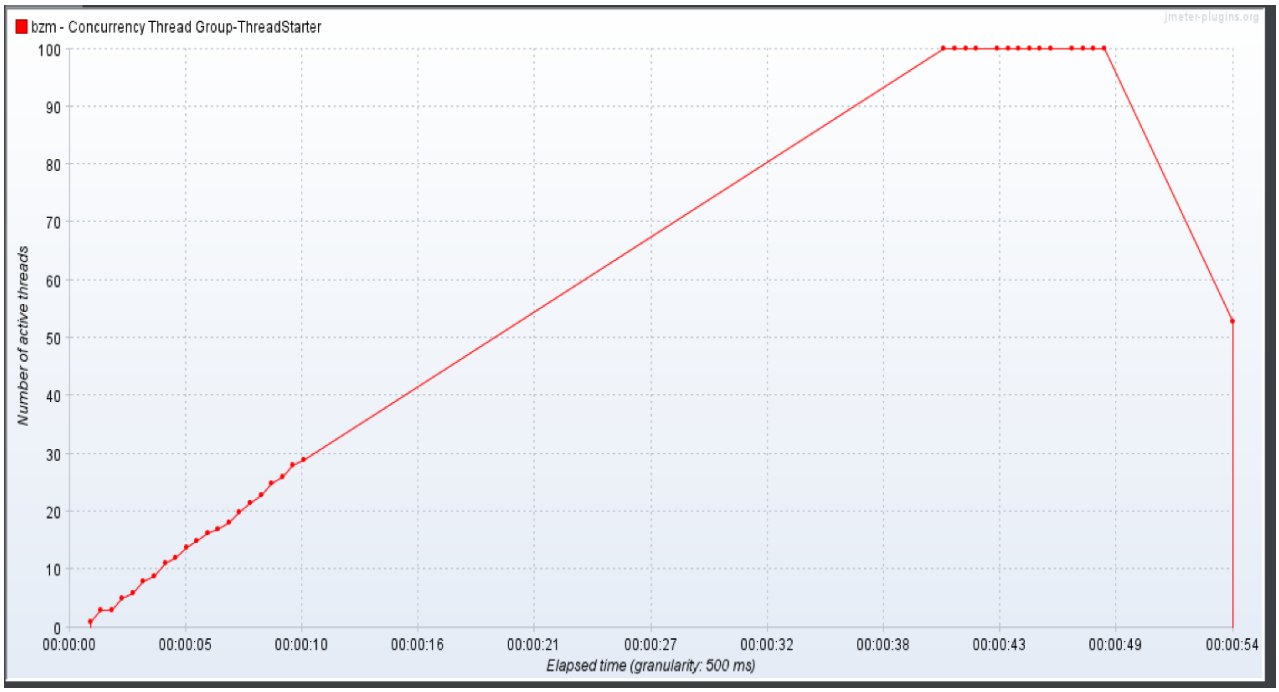
Summary Report



Response Times VS Time Graph



Active Threads VS Time Graph

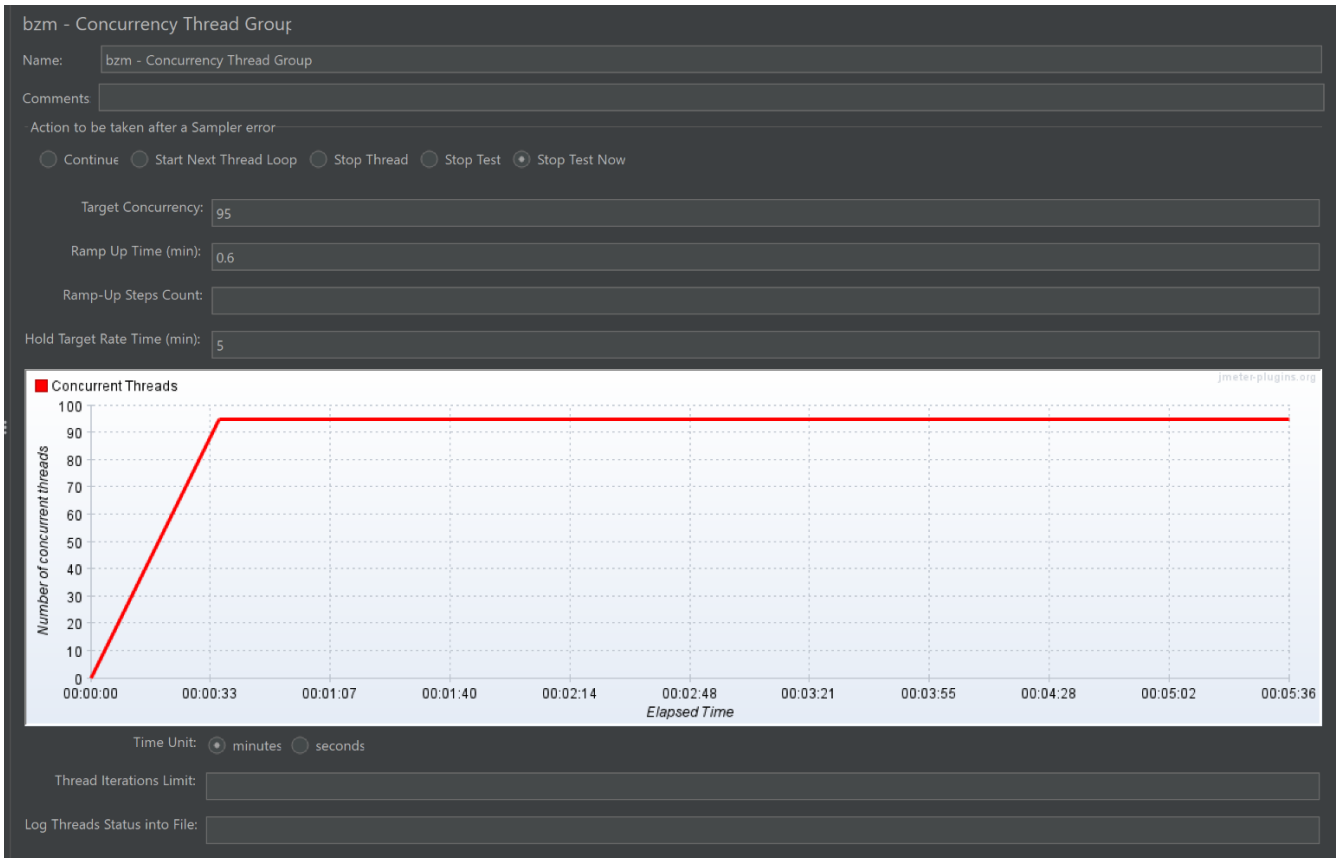


Result from test 3:
From the above graphs and reports, we can see that the system starts to throw errors once 100 concurrent users have reached. While checking manually, the response time is very high and the browser is not able to connect to the page and in order to connect several reloads are required.

This marks our upper limit of the system and now in order to validate if this is the true upper limit we decrease the number of concurrent threads by a little amount in the next test case.

Test Case 4:

Thread Configuration:



Summary Report

Summary Report

Name: Summary Report

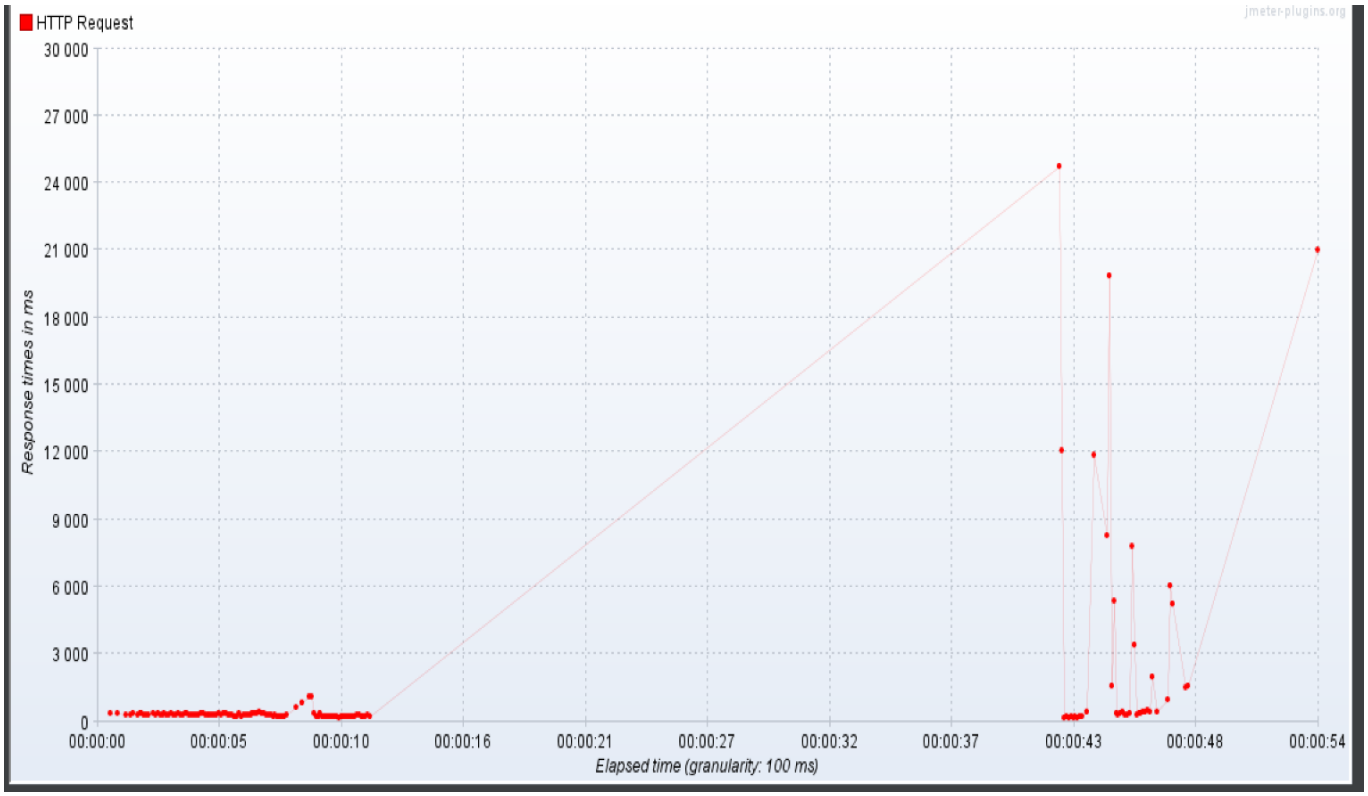
Comments

Write results to file / Read from file

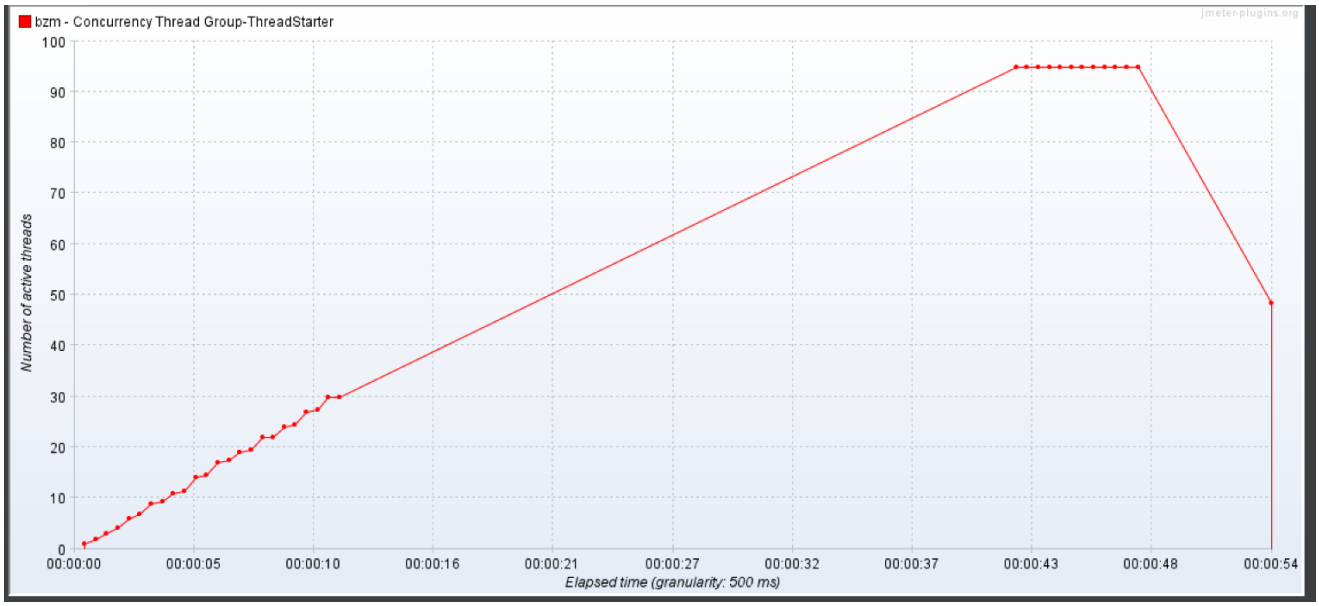
Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	1043	3230	134	42091	8821.31	9.11%	19.4/sec	59.47	2.20	3141.5
TOTAL	1043	3230	134	42091	8821.31	9.11%	19.4/sec	59.47	2.20	3141.5

Response Times VS Time Graph



Active Threads VS Time Graph



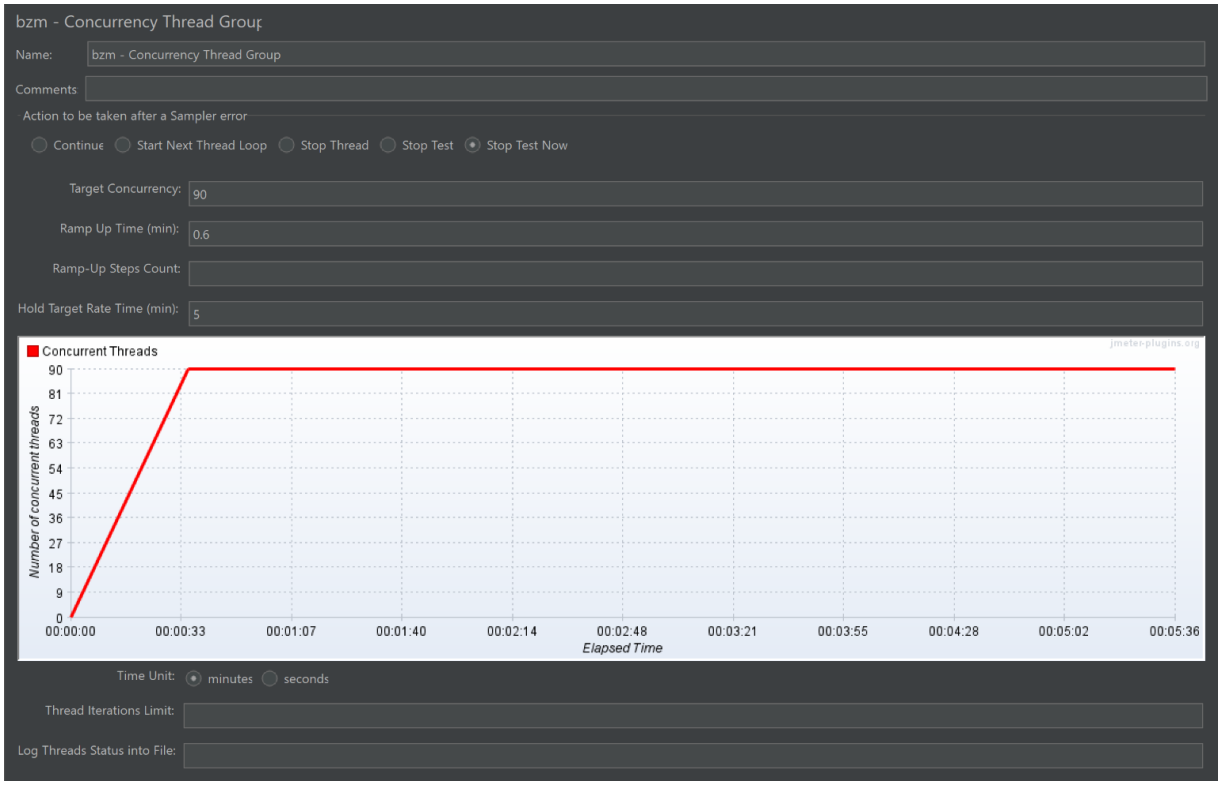
Result from test 4:

From the above graphs and reports, we can see that the system starts to throw errors once 95 concurrent users have reached. While checking manually, the response time is very high and the browser is not able to connect to the page and in order to connect several reloads are required.

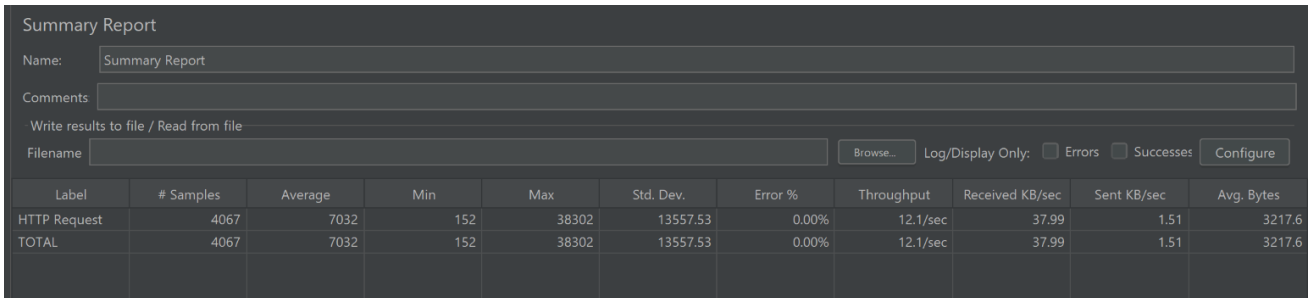
This marks our upper limit of the system and now in order to validate if this is the true upper limit we decrease the number of concurrent threads by a little amount in the next test case.

Test Case 5:

Thread Configuration:

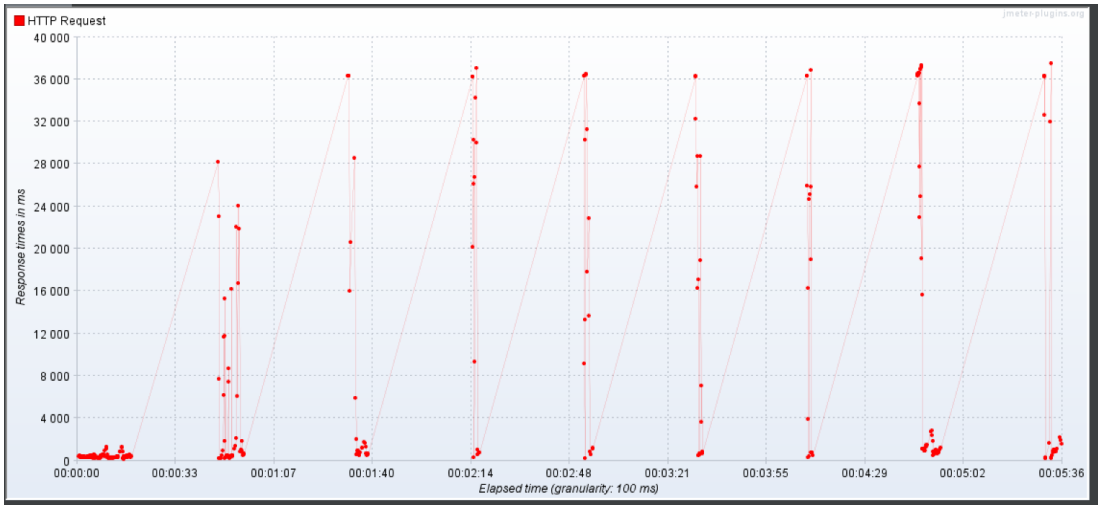


Summary Report

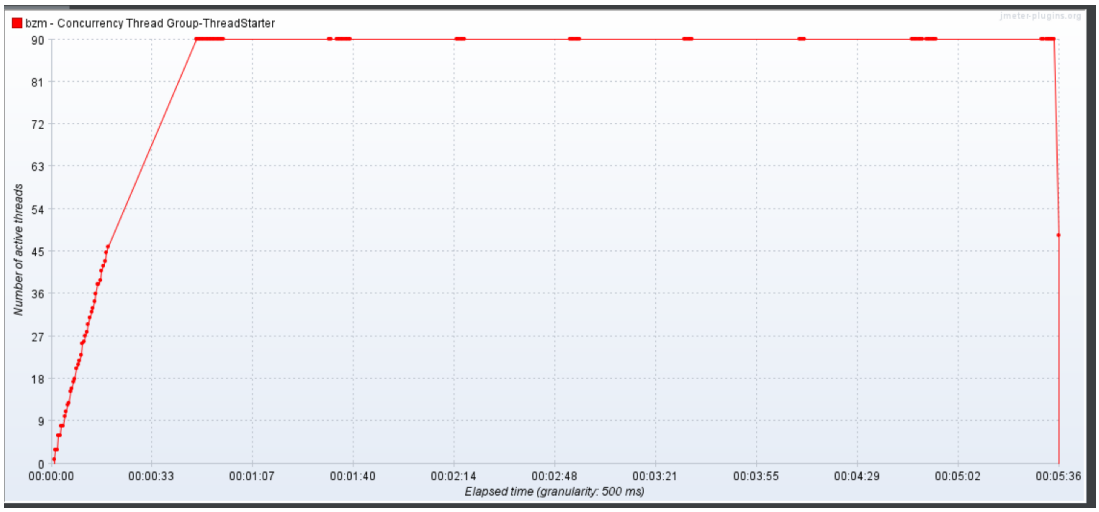


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	4067	7032	152	38302	13557.53	0.00%	12.1/sec	37.99	1.51	3217.6
TOTAL	4067	7032	152	38302	13557.53	0.00%	12.1/sec	37.99	1.51	3217.6

Response Times VS Time Graph



Active Threads VS Time Graph



Result of Test 5: From the above graphs and reports and the other test cases, it can be seen that the system is able to handle 90 users simultaneously without throwing errors.

Hence the stress limit of our system is 95 concurrent users and its capacity is 90 concurrent users.

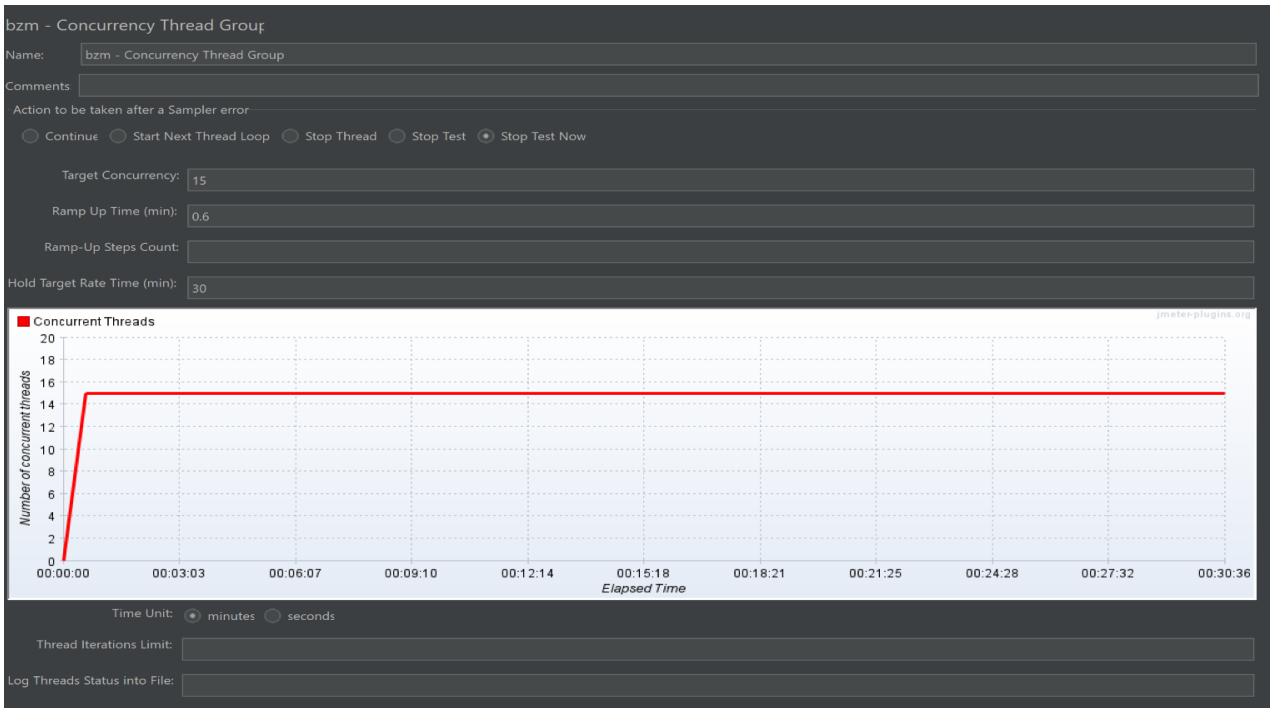
4. Endurance Test

Endurance tests help us to identify if our system is able to handle the optimal number of concurrent users over a long period of time within our set constraints.

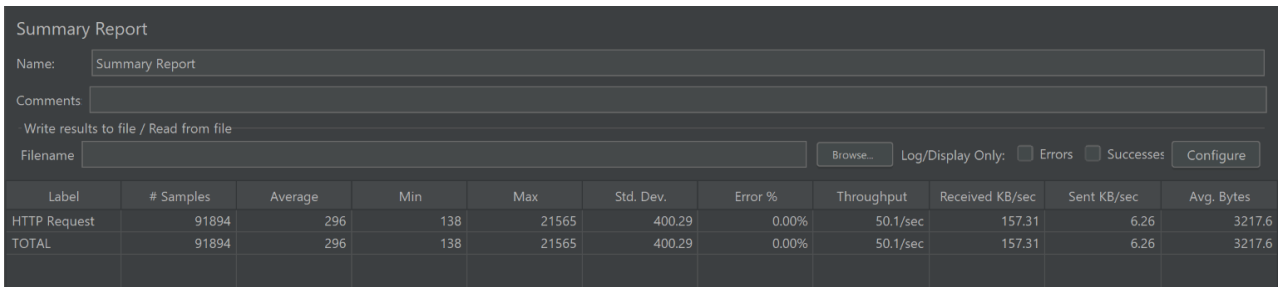
Note: While running the threads, the website was also tested manually to find if the test results were inline with the manual experience.

Test case 1

Thread Configuration:

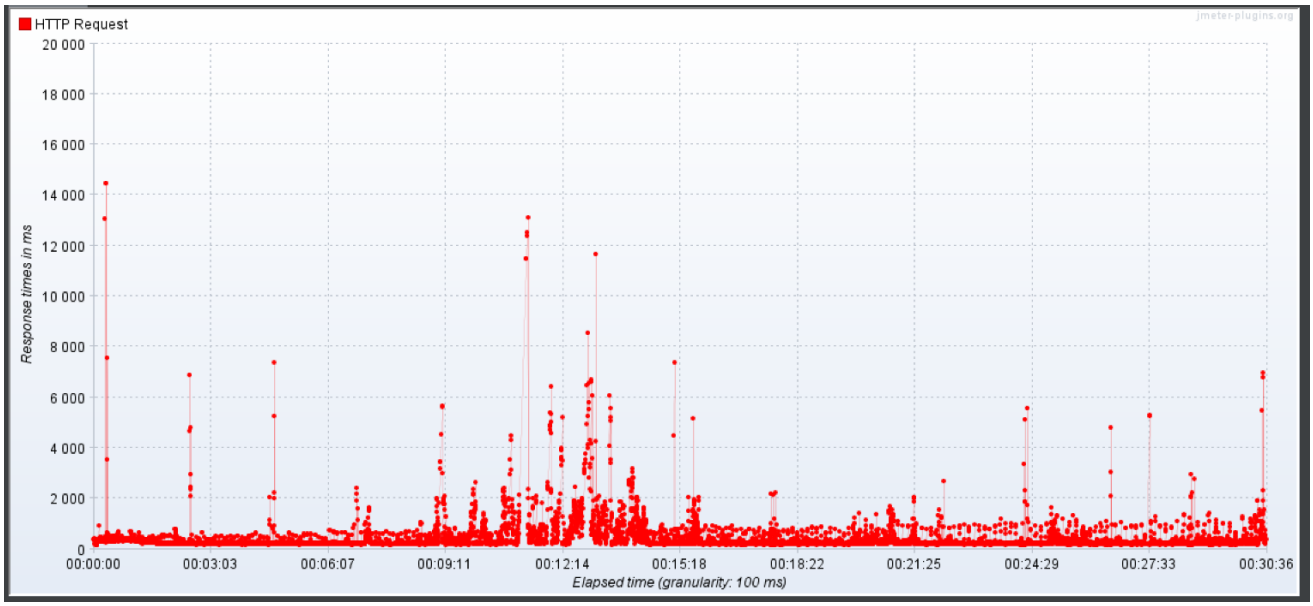


Summary Report:

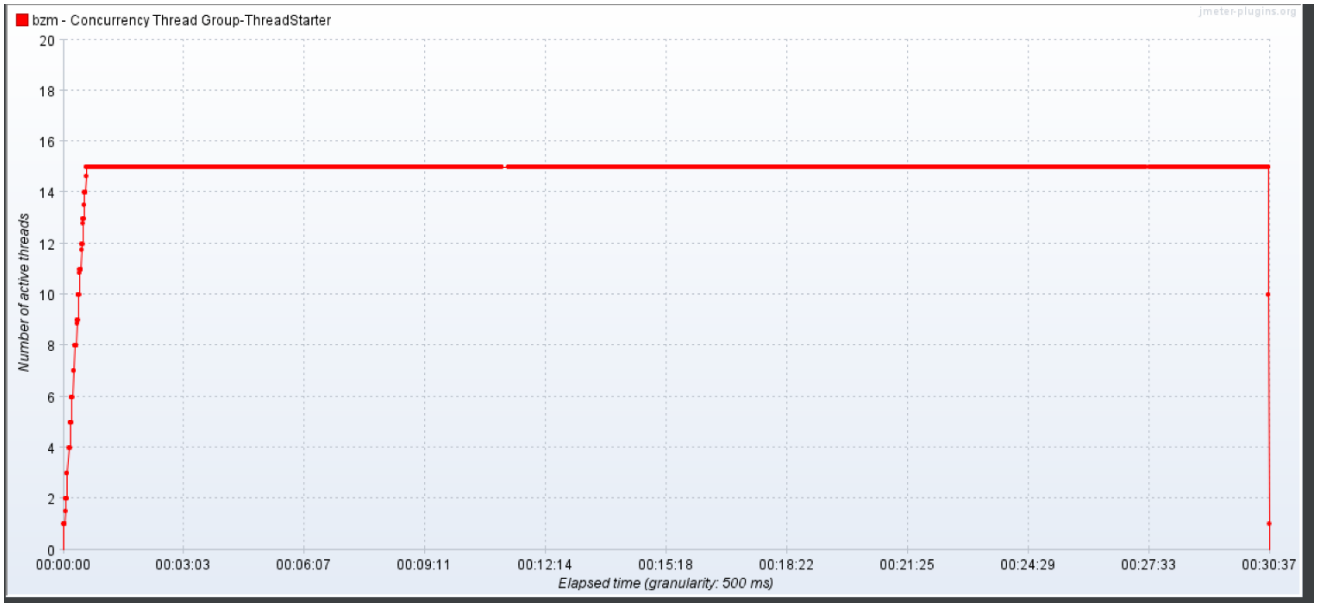


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	91894	296	138	21565	400.29	0.00%	50.1/sec	157.31	6.26	3217.6
TOTAL	91894	296	138	21565	400.29	0.00%	50.1/sec	157.31	6.26	3217.6

Response Times VS Time Graph



Active Threads VS Time Graph



From the performed test it can be seen that our system is able to handle the given load for an extended period of time although it suffers from some performance issues sometimes.