

Stress Performance Testing - Apache JMeter

Using Apache JMeter for the following stress performance tests.

Configuration

All stress test are configured to use this GET request for the deployed Munchora server.

The screenshot shows the 'HTTP Request' configuration dialog in Apache JMeter. The 'Name' field is set to 'GET Recipes'. The 'Basic' tab is selected. Under 'Web Server', the 'Protocol' is 'https', 'Server Name or IP' is 'munchora.pro', and 'Port Number' is empty. Under 'HTTP Request', the 'Method' is 'GET' and the 'Path' is '/api/v1/recipes'. The 'Content encoding' is empty. There are checkboxes for 'Redirect Automatically', 'Follow Redirects', 'Use KeepAlive' (checked), 'Use multipart/form-data', and 'Browser-compatible headers'. Below this, there are tabs for 'Parameters', 'Body Data', and 'Files Upload'. The 'Parameters' tab is active, showing a table with columns: Name, Value, URL Encode?, Content-Type, and Include Equals?.

Same assertions for the three different tests.

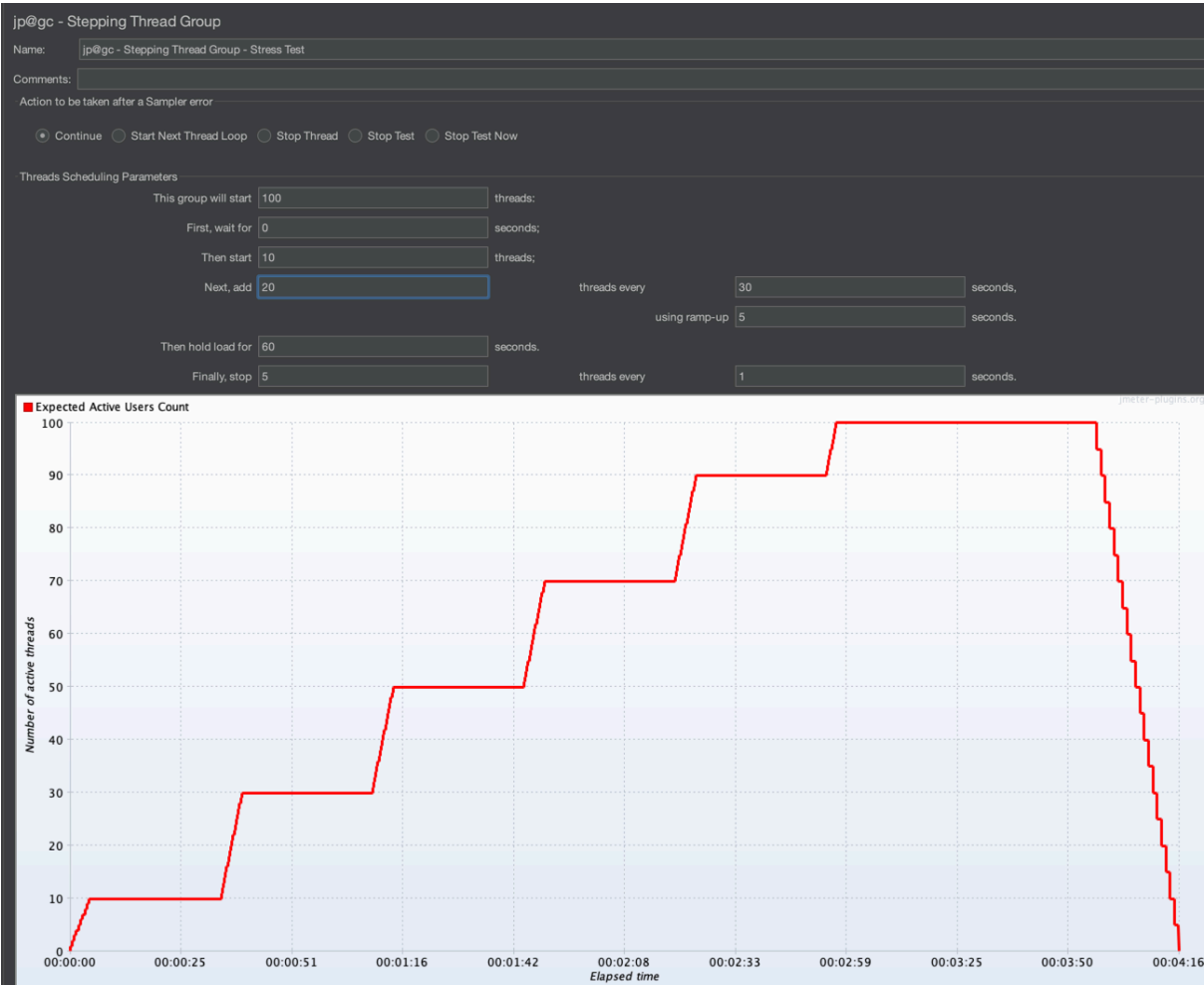
The screenshot shows the 'Response Assertion' configuration dialog in Apache JMeter. The 'Name' field is set to 'Response Assertion'. The 'Apply to:' section has 'Main sample only' selected. The 'Field to Test' section has 'Response Code' selected. The 'Pattern Matching Rules' section has 'Matches' selected and 'Or' checked. The 'Patterns to Test' table lists three patterns: 200, 302, and 503.

	Patterns to Test
1	200
1	302
1	503

- 200 → status OK
- 302 → Ruby on Rails rate limit has been exceeded - client gets redirected to <https://disney.com>
- 503 → Rate limit configured at Reverse Proxy Nginx.

Stress Testing

Push system beyond normal limits to find breaking points - makes use of extension
pg@gc – Stepping Thread Group .



Aggregated results:

Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename: ☐ Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
GET Recipes	19253	897	95	1111	3087	19092	43	195515	1.05%	47.3/sec	30.38	5.88
TOTAL	19253	897	95	1111	3087	19092	43	195515	1.05%	47.3/sec	30.38	5.88

Load testing

Verify system performance under expected normal-to-peak load (the implementation is expecting middle input)

Thread group setting	Value	Rationale
Number of Threads	200	Expected peak concurrent users
Ramp-up Period	20 seconds	Smooth, gradual increase
Loop Count	5	Enough to gather statistics

Aggregated results:

Aggregate Report

Name:Aggregate Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only: ☐ Errors ☐ Successes

Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
GET Recipes	1000	90	87	96	105	169	74	257	0.00%	49.1/sec	68.54	6.13
TOTAL	1000	90	87	96	105	169	74	257	0.00%	49.1/sec	68.54	6.13

Spike testing

Verify system performance under a suddenly huge amount of traffic towards server.

Thread group setting	Value	Rationale
Number of Threads	1000 (A huge number)	Simulates the peak of the surge
Ramp-up Period	0 seconds	Very short to simulate a sudden spike
Loop Count	10	Load hits once and immediately stops

Aggregated results:

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
GET Recipes	5680	2827	137	57628	4951.33	0.02%	98.4/sec	52.88	12.30	550.1
TOTAL	5680	2827	137	57628	4951.33	0.02%	98.4/sec	52.88	12.30	550.1

Test system response to sudden, massive traffic increases.

```
# Ensure to be positioned at ./stress-performance-testing/
# run stress tests
jmeter -n -t stepping_thread_group_stress_test.jmx -l results/load_results.jtl
-e -o reports/load_test

# run load tests
jmeter -n -t load_test.jmx -l results/load_results.jtl -e -o reports/load_test

# run spike tests
jmeter -n -t spike_test.jmx -l results/load_results.jtl -e -o
reports/load_test
```