Iteration 3

Test Records

1 Number of Users Stress Testing:

1. 200 users x 500 requests per user (Run 10 minutes)
    a. Configuration:

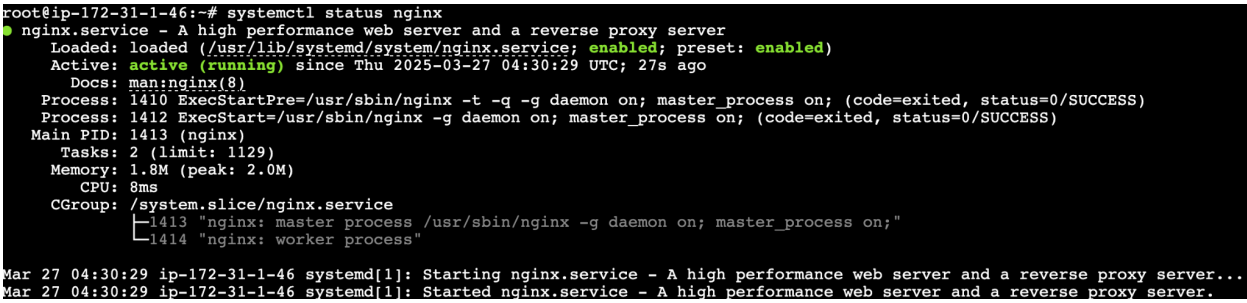
```
root@ip-172-31-15-56:~/test-0326# cat test-user-stress-v2.jmx
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
        <TestPlan guiclass="TestPlanGui" testclass="TestPlan"
testname="Enhanced User Stress Test Plan" enabled="true">
        <stringProp name="TestPlan.comments">Testing system
performance under 200 concurrent users with comprehensive
reporting</stringProp>
        <boolProp name="TestPlan.functional_mode">false</boolProp>
        <boolProp
name="TestPlan.serialize_threadgroups">false</boolProp>
        <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments" guiclass="ArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
        <stringProp name="Argument.name">HOST</stringProp>
        <stringProp name="Argument.value">172.31.1.46</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
        <stringProp name="Argument.name">PORT</stringProp>
        <stringProp name="Argument.value">80</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
        <stringProp name="Argument.name">PROTOCOL</stringProp>
        <stringProp name="Argument.value">http</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
```

```
        </elementProp>
        </collectionProp>
        </elementProp>
        <stringProp name="TestPlan.user_define_classpath"></stringProp>
        </TestPlan>
        <hashTree>
        <ThreadGroup guiclass="ThreadGroupGui"
testclass="ThreadGroup" testname="User Stress Test Group"
enabled="true">
        <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp
name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
        </elementProp>
        <stringProp name="ThreadGroup.num_threads">200</stringProp>
        <stringProp name="ThreadGroup.ramp_time">30</stringProp>
        <boolProp name="ThreadGroup.scheduler">true</boolProp>
        <stringProp name="ThreadGroup.duration">600</stringProp>
        <stringProp name="ThreadGroup.delay">5</stringProp>
        </ThreadGroup>
        <hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="HTTP Request"
enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp
name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/</stringProp>
```

```
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp
name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp
name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp
name="HTTPSampler.response_timeout">30000</stringProp>
        </HTTPSamplerProxy>
        <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Response Code Assertion"
enabled="true">
        <collectionProp name="Asserion.test_strings">
        <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time Assertion"
enabled="true">
        <stringProp name="DurationAssertion.duration">1000</stringProp>
        </DurationAssertion>
        <hashTree/>
        </hashTree>
        <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Realistic User Timing"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
        </UniformRandomTimer>
```

```
        <hashTree/>
        </hashTree>
        </hashTree>
  </hashTree>
</jmeterTestPlan>
```

b. Test:
   i. Check Nginx status - OK



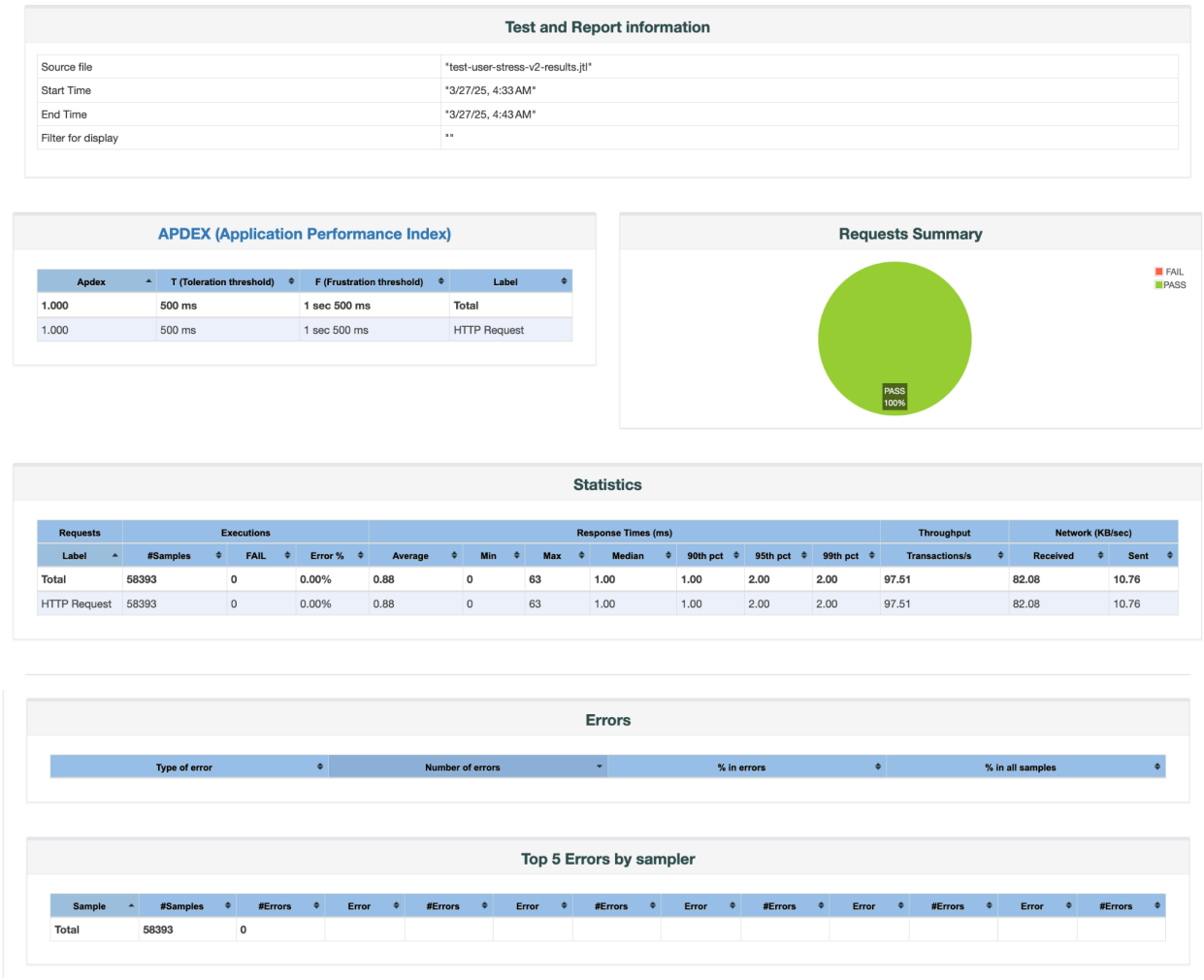
   ii. Run a test using JMeter: jmeter -n -t test-user-stress-v2.jmx -l test-user-stress-v2-results.jtl -e -o ./report

c. Result:

```
root@ip-172-31-15-56:~/test-0326# jmeter -n -t test-user-stress-v2.jmx -l
test-user-stress-v2-results.jtl -e -o ./report
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-user-stress-v2.jmx
Starting standalone test @ 2025 Mar 27 04:33:11 UTC (1743049991708)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
message on port 4445
summary +    244 in 00:00:18 =   13.6/s Avg:     2 Min:           0 Max:
63 Err:          0 (0.00%) Active: 86 Started: 86 Finished: 0
summary +   2459 in 00:00:30 =   82.0/s Avg:     1 Min:           0 Max:
21 Err:          0 (0.00%) Active: 200 Started: 200 Finished: 0
```

summary = 2703 in 00:00:48 = 56.5/s Avg: 1 Min: 0 Max: 63 Err: 0 (0.00%
summary + 3007 in 00:00:30 = 100.2/s Avg: 0 Min: 0 Max: 10 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 5710 in 00:01:18 = 73.3/s Avg: 1 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 3003 in 00:00:30 = 100.1/s Avg: 0 Min: 0 Max: 10 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 8713 in 00:01:48 = 80.8/s Avg: 1 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 3000 in 00:00:30 = 100.1/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 11713 in 00:02:18 = 85.0/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 3011 in 00:00:30 = 100.4/s Avg: 0 Min: 0 Max: 9 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 14724 in 00:02:48 = 87.7/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2984 in 00:00:30 = 99.5/s Avg: 0 Min: 0 Max: 4 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 17708 in 00:03:18 = 89.5/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2990 in 00:00:30 = 99.7/s Avg: 0 Min: 0 Max: 10 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 20698 in 00:03:48 = 90.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2980 in 00:00:30 = 99.3/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 23678 in 00:04:18 = 91.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 3008 in 00:00:30 = 100.3/s Avg: 0 Min: 0 Max: 11 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 26686 in 00:04:48 = 92.7/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2999 in 00:00:30 = 100.0/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 29685 in 00:05:18 = 93.4/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 3022 in 00:00:30 = 100.6/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0

summary = 32707 in 00:05:48 = 94.0/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2994 in 00:00:30 = 99.9/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 35701 in 00:06:18 = 94.5/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 3012 in 00:00:30 = 100.4/s Avg: 0 Min: 0 Max: 4 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 38713 in 00:06:48 = 94.9/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 3021 in 00:00:30 = 100.6/s Avg: 0 Min: 0 Max: 8 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 41734 in 00:07:18 = 95.3/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 3016 in 00:00:30 = 100.6/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 44750 in 00:07:48 = 95.6/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2976 in 00:00:30 = 99.1/s Avg: 0 Min: 0 Max: 7 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 47726 in 00:08:18 = 95.9/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2993 in 00:00:30 = 99.8/s Avg: 0 Min: 0 Max: 4 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 50719 in 00:08:48 = 96.1/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2974 in 00:00:30 = 99.1/s Avg: 0 Min: 0 Max: 8 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 53693 in 00:09:18 = 96.2/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 2984 in 00:00:30 = 99.5/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 0
summary = 56677 in 00:09:48 = 96.4/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 1716 in 00:00:17 = 98.8/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 0 Started: 200 Finished: 200
summary = 58393 in 00:10:05 = 96.5/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 27 04:43:17 UTC (1743050597364)
... end of run



Test and Report information

Source file	"test-user-stress-v2-results.jtl"
Start Time	"3/27/25, 4:33 AM"
End Time	"3/27/25, 4:43 AM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	500 ms	1 sec 500 ms	Total
1.000	500 ms	1 sec 500 ms	HTTP Request

Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	58393	0	0.00%	0.88	0	63	1.00	1.00	2.00	2.00	97.51	82.08	10.76
HTTP Request	58393	0	0.00%	0.88	0	63	1.00	1.00	2.00	2.00	97.51	82.08	10.76

Errors

Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	58393	0										

2. 1000 users x 500 requests per user (10 minutes)
    a. Configuration:
    root@ip-172-31-15-56:~/test-0326# cat test-user-stress-v2-1000.jmx
    <?xml version="1.0" encoding="UTF-8"?>
    <jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
      <hashTree>
        <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Enhanced User Stress Test Plan" enabled="true">
        <stringProp name="TestPlan.comments">Testing system performance under 1000 concurrent users with comprehensive reporting</stringProp>
        <boolProp name="TestPlan.functional_mode">false</boolProp>

```
        <boolProp
name="TestPlan.serialize_threadgroups">false</boolProp>
        <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments" guiclass="ArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
        <stringProp name="Argument.name">HOST</stringProp>
        <stringProp name="Argument.value">172.31.1.46</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
        <stringProp name="Argument.name">PORT</stringProp>
        <stringProp name="Argument.value">80</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
        <stringProp name="Argument.name">PROTOCOL</stringProp>
        <stringProp name="Argument.value">http</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        </collectionProp>
        </elementProp>
        <stringProp name="TestPlan.user_define_classpath"></stringProp>
        </TestPlan>
        <hashTree>
        <ThreadGroup guiclass="ThreadGroupGui"
testclass="ThreadGroup" testname="User Stress Test Group"
enabled="true">
        <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp
name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
        </elementProp>
```

```
        <stringProp
name="ThreadGroup.num_threads">1000</stringProp>
        <stringProp name="ThreadGroup.ramp_time">60</stringProp>
        <boolProp name="ThreadGroup.scheduler">true</boolProp>
        <stringProp name="ThreadGroup.duration">600</stringProp>
        <stringProp name="ThreadGroup.delay">5</stringProp>
        </ThreadGroup>
        <hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="HTTP Request"
enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp
name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp
name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp
name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp
name="HTTPSampler.response_timeout">30000</stringProp>
        </HTTPSamplerProxy>
        <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Response Code Assertion"
enabled="true">
```

```
        <collectionProp name="Asserion.test_strings">
        <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time Assertion"
enabled="true">
        <stringProp name="DurationAssertion.duration">1000</stringProp>
        </DurationAssertion>
        <hashTree/>
        </hashTree>
        <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Realistic User Timing"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
        </UniformRandomTimer>
        <hashTree/>
        </hashTree>
        </hashTree>
  </hashTree>
</jmeterTestPlan>
```

b. Test:
   jmeter -n -t test-user-stress-v2-1000.jmx -l
   test-user-stress-v2-1000-results.jtl -e -o ./test-user-stress-v2-1000-report

c. Result:
   root@ip-172-31-15-56:~/test-0326# jmeter -n -t
   test-user-stress-v2-1000.jmx -l test-user-stress-v2-1000-results.jtl -e -o
   ./test-user-stress-v2-1000-report
   WARN StatusConsoleListener The use of package scanning to locate
   plugins is deprecated and will be removed in a future release

WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-user-stress-v2-1000.jmx
Starting standalone test @ 2025 Mar 27 05:10:11 UTC (1743052211952)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 572 in 00:00:18 = 32.4/s Avg: 1 Min: 0 Max: 85 Err: 0 (0.00%) Active: 211 Started: 211 Finished: 0
summary + 6659 in 00:00:30 = 222.1/s Avg: 0 Min: 0 Max: 23 Err: 0 (0.00%) Active: 710 Started: 710 Finished: 0
summary = 7231 in 00:00:48 = 151.8/s Avg: 1 Min: 0 Max: 85 Err: 0 (0.00%)
summary + 13569 in 00:00:30 = 452.3/s Avg: 0 Min: 0 Max: 53 Err: 2667 (19.66%) Active: 1000 Started: 1000 Finished: 0
summary = 20800 in 00:01:18 = 267.9/s Avg: 0 Min: 0 Max: 85 Err: 2667 (12.82%)
summary + 15045 in 00:00:30 = 501.5/s Avg: 0 Min: 0 Max: 39 Err: 3830 (25.46%) Active: 1000 Started: 1000 Finished: 0
summary = 35845 in 00:01:48 = 333.0/s Avg: 0 Min: 0 Max: 85 Err: 6497 (18.13%)
summary + 14979 in 00:00:30 = 499.3/s Avg: 0 Min: 0 Max: 28 Err: 3777 (25.22%) Active: 1000 Started: 1000 Finished: 0
summary = 50824 in 00:02:18 = 369.3/s Avg: 0 Min: 0 Max: 85 Err: 10274 (20.21%)
summary + 15035 in 00:00:30 = 501.1/s Avg: 0 Min: 0 Max: 29 Err: 3820 (25.41%) Active: 1000 Started: 1000 Finished: 0
summary = 65859 in 00:02:48 = 392.8/s Avg: 0 Min: 0 Max: 85 Err: 14094 (21.40%)
summary + 14991 in 00:00:30 = 499.8/s Avg: 0 Min: 0 Max: 31 Err: 3826 (25.52%) Active: 1000 Started: 1000 Finished: 0
summary = 80850 in 00:03:18 = 409.1/s Avg: 0 Min: 0 Max: 85 Err: 17920 (22.16%)
summary + 15053 in 00:00:30 = 501.7/s Avg: 0 Min: 0 Max: 32 Err: 3833 (25.46%) Active: 1000 Started: 1000 Finished: 0

summary = 95903 in 00:03:48 = 421.3/s Avg: 0 Min: 0 Max: 85 Err: 21753 (22.68%)
summary + 15014 in 00:00:30 = 500.5/s Avg: 0 Min: 0 Max: 27 Err: 3808 (25.36%) Active: 1000 Started: 1000 Finished: 0
summary = 110917 in 00:04:18 = 430.5/s Avg: 0 Min: 0 Max: 85 Err: 25561 (23.05%)
summary + 14957 in 00:00:30 = 498.6/s Avg: 0 Min: 0 Max: 31 Err: 3791 (25.35%) Active: 1000 Started: 1000 Finished: 0
summary = 125874 in 00:04:48 = 437.6/s Avg: 0 Min: 0 Max: 85 Err: 29352 (23.32%)
summary + 15000 in 00:00:30 = 499.9/s Avg: 0 Min: 0 Max: 29 Err: 3777 (25.18%) Active: 1000 Started: 1000 Finished: 0
summary = 140874 in 00:05:18 = 443.5/s Avg: 0 Min: 0 Max: 85 Err: 33129 (23.52%)
summary + 14982 in 00:00:30 = 499.5/s Avg: 0 Min: 0 Max: 26 Err: 3802 (25.38%) Active: 1000 Started: 1000 Finished: 0
summary = 155856 in 00:05:48 = 448.3/s Avg: 0 Min: 0 Max: 85 Err: 36931 (23.70%)
summary + 14958 in 00:00:30 = 498.6/s Avg: 0 Min: 0 Max: 28 Err: 3836 (25.65%) Active: 1000 Started: 1000 Finished: 0
summary = 170814 in 00:06:18 = 452.3/s Avg: 0 Min: 0 Max: 85 Err: 40767 (23.87%)
summary + 15014 in 00:00:30 = 500.5/s Avg: 0 Min: 0 Max: 28 Err: 3784 (25.20%) Active: 1000 Started: 1000 Finished: 0
summary = 185828 in 00:06:48 = 455.9/s Avg: 0 Min: 0 Max: 85 Err: 44551 (23.97%)
summary + 15017 in 00:00:30 = 500.5/s Avg: 0 Min: 0 Max: 28 Err: 3804 (25.33%) Active: 1000 Started: 1000 Finished: 0
summary = 200845 in 00:07:18 = 458.9/s Avg: 0 Min: 0 Max: 85 Err: 48355 (24.08%)
summary + 14945 in 00:00:30 = 498.2/s Avg: 0 Min: 0 Max: 30 Err: 3800 (25.43%) Active: 1000 Started: 1000 Finished: 0
summary = 215790 in 00:07:48 = 461.4/s Avg: 0 Min: 0 Max: 85 Err: 52155 (24.17%)
summary + 14961 in 00:00:30 = 498.7/s Avg: 0 Min: 0 Max: 29 Err: 3812 (25.48%) Active: 1000 Started: 1000 Finished: 0
summary = 230751 in 00:08:18 = 463.7/s Avg: 0 Min: 0 Max: 85 Err: 55967 (24.25%)
summary + 14974 in 00:00:30 = 499.1/s Avg: 0 Min: 0 Max: 31 Err: 3806 (25.42%) Active: 1000 Started: 1000 Finished: 0

summary = 245725 in 00:08:48 = 465.7/s Avg: 0 Min: 0 Max: 85 Err: 59773 (24.33%)
summary + 15019 in 00:00:30 = 500.7/s Avg: 0 Min: 0 Max: 29 Err: 3851 (25.64%) Active: 1000 Started: 1000 Finished: 0
summary = 260744 in 00:09:18 = 467.6/s Avg: 0 Min: 0 Max: 85 Err: 63624 (24.40%)
summary + 14950 in 00:00:30 = 498.3/s Avg: 0 Min: 0 Max: 27 Err: 3796 (25.39%) Active: 1000 Started: 1000 Finished: 0
summary = 275694 in 00:09:48 = 469.2/s Avg: 0 Min: 0 Max: 85 Err: 67420 (24.45%)
summary + 8895 in 00:00:18 = 491.5/s Avg: 0 Min: 0 Max: 29 Err: 1950 (21.92%) Active: 0 Started: 1000 Finished: 1000
summary = 284589 in 00:10:06 = 469.8/s Avg: 0 Min: 0 Max: 85 Err: 69370 (24.38%)
Tidying up ... @ 2025 Mar 27 05:20:18 UTC (1743052818100)
... end of run

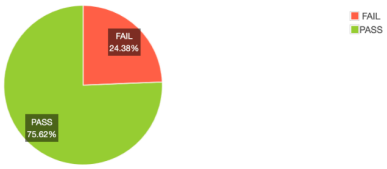
Test and Report information

Source file	"test-user-stress-v2-1000-results.jtl"
Start Time	"3/27/25, 5:10 AM"
End Time	"3/27/25, 5:20 AM"
Filter for display	""

APDEX (Application Performance Index)

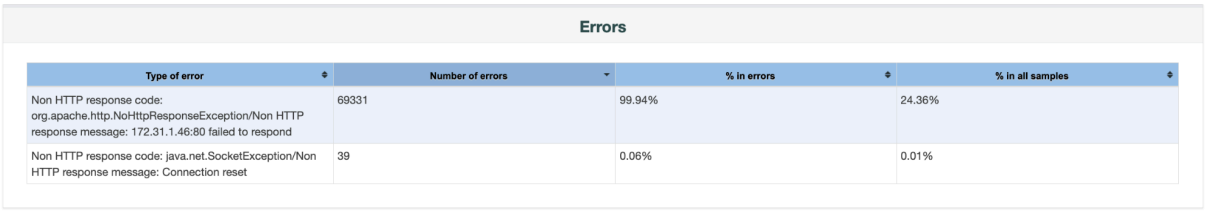
Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.756	500 ms	1 sec 500 ms	Total
0.756	500 ms	1 sec 500 ms	HTTP Request

Requests Summary



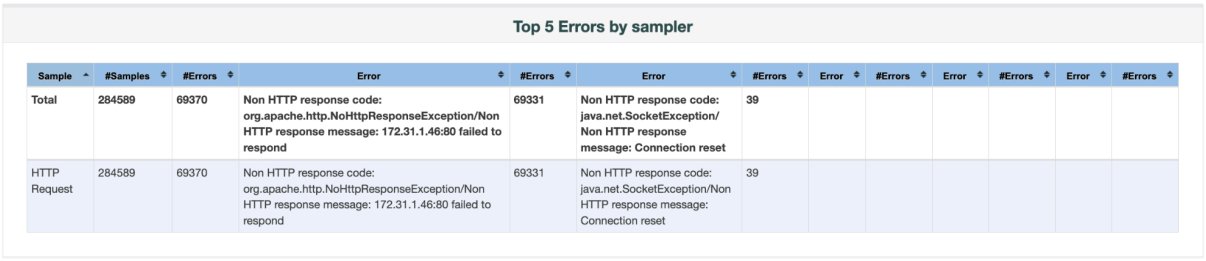
Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	284589	69370	24.38%	0.71	0	85	1.00	1.00	2.00	3.00	474.91	536.15	39.63
HTTP Request	284589	69370	24.38%	0.71	0	85	1.00	1.00	2.00	3.00	474.91	536.15	39.63



Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 172.31.1.46:80 failed to respond	69331	99.94%	24.36%
Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	39	0.06%	0.01%

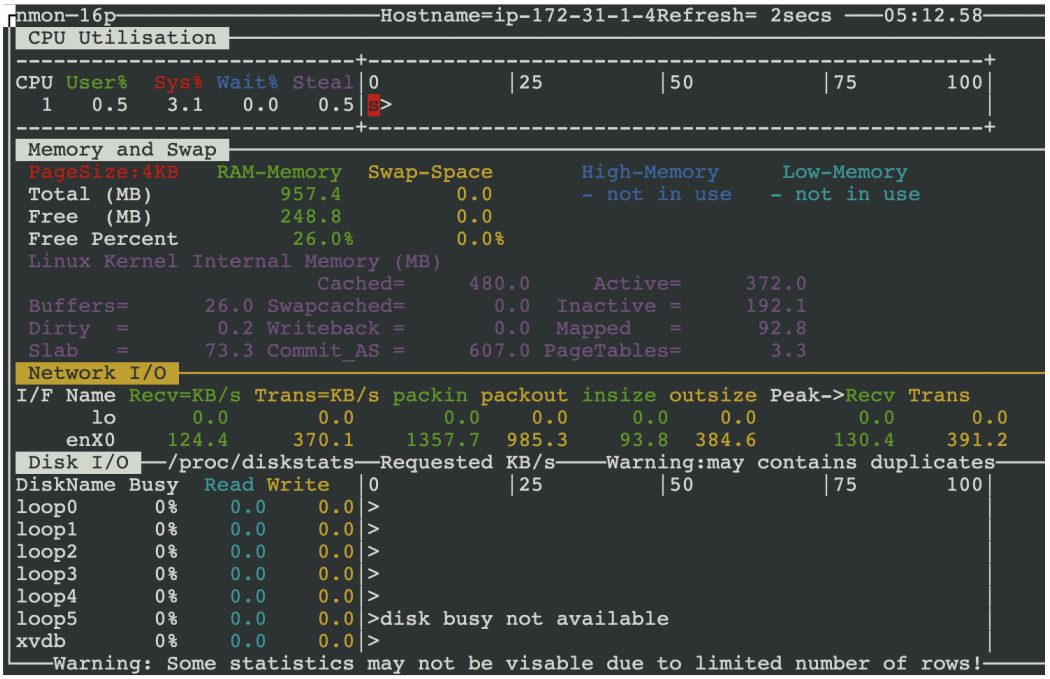


Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	**284589**	**69370**	**Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 172.31.1.46:80 failed to respond**	**69331**	**Non HTTP response code: java.net.SocketException/ Non HTTP response message: Connection reset**	**39**						
HTTP Request	284589	69370	Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 172.31.1.46:80 failed to respond	69331	Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	39						

d. Result analysis:
    i. Response times remained good, but the system couldn't handle all requests. Total error count: 69,370 (24.38% of all requests). Overall average of 469.8 req/sec..
    Bottleneck Found: With 1000 concurrent users, the system consistently rejects ~25% of requests.
    ii. Potential error causes:
        - **Nginx Worker Connections**: `worker_connections 768;`
            a. This limits Nginx to handling at most 768 simultaneous connections
            b. With 1000 concurrent users in your test, we re exceeding this limit
        - **System File Descriptor Limit**: `ulimit -n 1024`
            a. Each connection uses a file descriptor
            b. Our system limit is 1024, which is only slightly higher than Our user count
            c. Some file descriptors are used for other purposes (logs, etc.), leaving fewer for connections
        - **TCP Backlog Setting**: `net.ipv4.tcp_max_syn_backlog = 128`
            a. This limits the queue of incoming connection requests
            b. With 1000 users connecting rapidly, this queue fills up quickly

- CPU & memory does not seem to be the limiting factors:



i-09eac4388dcf36619 (Nginx Server1)

PublicIPs: 52.53.164.74 PrivateIPs: 172.31.1.46

3. 768 users x 500 requests per user (10 min)
    a. Configuration:
    root@ip-172-31-15-56:~/test-0326# cat test-user-stress-v2-768.jmx
    <?xml version="1.0" encoding="UTF-8"?>
    <jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
      <hashTree>
        <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Enhanced User Stress Test Plan" enabled="true">
        <stringProp name="TestPlan.comments">Testing system performance under 768 concurrent users with comprehensive reporting</stringProp>
        <boolProp name="TestPlan.functional_mode">false</boolProp>
        <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
        <elementProp name="TestPlan.user_defined_variables" elementType="Arguments" guiclass="ArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">

```
        <elementProp name="HOST" elementType="Argument">
        <stringProp name="Argument.name">HOST</stringProp>
        <stringProp name="Argument.value">172.31.1.46</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
        <stringProp name="Argument.name">PORT</stringProp>
        <stringProp name="Argument.value">80</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
        <stringProp name="Argument.name">PROTOCOL</stringProp>
        <stringProp name="Argument.value">http</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        </collectionProp>
        </elementProp>
        <stringProp name="TestPlan.user_define_classpath"></stringProp>
        </TestPlan>
        <hashTree>
        <ThreadGroup guiclass="ThreadGroupGui"
testclass="ThreadGroup" testname="User Stress Test Group"
enabled="true">
        <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp
name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
        </elementProp>
        <stringProp name="ThreadGroup.num_threads">768</stringProp>
        <stringProp name="ThreadGroup.ramp_time">30</stringProp>
        <boolProp name="ThreadGroup.scheduler">true</boolProp>
        <stringProp name="ThreadGroup.duration">600</stringProp>
        <stringProp name="ThreadGroup.delay">5</stringProp>
        </ThreadGroup>
        <hashTree>
```

```
        <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="HTTP Request"
enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp
name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp
name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp
name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp
name="HTTPSampler.response_timeout">30000</stringProp>
        </HTTPSamplerProxy>
        <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Response Code Assertion"
enabled="true">
        <collectionProp name="Asserion.test_strings">
        <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp name="Assertion.custom_message"></stringProp>
        <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
```

```
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time Assertion"
enabled="true">
        <stringProp name="DurationAssertion.duration">1000</stringProp>
        </DurationAssertion>
        <hashTree/>
        </hashTree>
        <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Realistic User Timing"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
        </UniformRandomTimer>
        <hashTree/>
        </hashTree>
        </hashTree>
  </hashTree>
</jmeterTestPlan>
```

b. Test:
   jmeter -n -t test-user-stress-v2-768.jmx -l test-user-stress-v2-768-results.jtl -e -o ./test-user-stress-v2-768-report
c. Result:
   root@ip-172-31-15-56:~/test-0326# jmeter -n -t test-user-stress-v2-768.jmx -l test-user-stress-v2-768-results.jtl -e -o ./test-user-stress-v2-768-report
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   Creating summariser <summary>
   Created the tree successfully using test-user-stress-v2-768.jmx
   Starting standalone test @ 2025 Mar 27 05:47:38 UTC (1743054458931)

Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 1381 in 00:00:21 = 66.8/s Avg: 1 Min: 0 Max: 74 Err: 0 (0.00%) Active: 401 Started: 401 Finished: 0
summary + 10037 in 00:00:30 = 334.6/s Avg: 0 Min: 0 Max: 33 Err: 480 (4.78%) Active: 768 Started: 768 Finished: 0
summary = 11418 in 00:00:51 = 225.4/s Avg: 0 Min: 0 Max: 74 Err: 480 (4.20%)
summary + 11489 in 00:00:30 = 383.0/s Avg: 0 Min: 0 Max: 56 Err: 892 (7.76%) Active: 768 Started: 768 Finished: 0
summary = 22907 in 00:01:21 = 284.0/s Avg: 0 Min: 0 Max: 74 Err: 1372 (5.99%)
summary + 11552 in 00:00:30 = 385.1/s Avg: 0 Min: 0 Max: 26 Err: 918 (7.95%) Active: 768 Started: 768 Finished: 0
summary = 34459 in 00:01:51 = 311.4/s Avg: 0 Min: 0 Max: 74 Err: 2290 (6.65%)
summary + 11481 in 00:00:30 = 382.7/s Avg: 0 Min: 0 Max: 21 Err: 920 (8.01%) Active: 768 Started: 768 Finished: 0
summary = 45940 in 00:02:21 = 326.6/s Avg: 0 Min: 0 Max: 74 Err: 3210 (6.99%)
summary + 11511 in 00:00:30 = 383.7/s Avg: 0 Min: 0 Max: 22 Err: 909 (7.90%) Active: 768 Started: 768 Finished: 0
summary = 57451 in 00:02:51 = 336.6/s Avg: 0 Min: 0 Max: 74 Err: 4119 (7.17%)
summary + 11500 in 00:00:30 = 383.4/s Avg: 0 Min: 0 Max: 27 Err: 898 (7.81%) Active: 768 Started: 768 Finished: 0
summary = 68951 in 00:03:21 = 343.6/s Avg: 0 Min: 0 Max: 74 Err: 5017 (7.28%)
summary + 11494 in 00:00:30 = 383.1/s Avg: 0 Min: 0 Max: 19 Err: 893 (7.77%) Active: 768 Started: 768 Finished: 0
summary = 80445 in 00:03:51 = 348.8/s Avg: 0 Min: 0 Max: 74 Err: 5910 (7.35%)
summary + 11516 in 00:00:30 = 383.8/s Avg: 0 Min: 0 Max: 21 Err: 907 (7.88%) Active: 768 Started: 768 Finished: 0
summary = 91961 in 00:04:21 = 352.8/s Avg: 0 Min: 0 Max: 74 Err: 6817 (7.41%)
summary + 11504 in 00:00:30 = 383.5/s Avg: 0 Min: 0 Max: 23 Err: 901 (7.83%) Active: 768 Started: 768 Finished: 0
summary = 103465 in 00:04:51 = 356.0/s Avg: 0 Min: 0 Max: 74 Err: 7718 (7.46%)

summary + 11468 in 00:00:30 = 382.3/s Avg: 0 Min: 0 Max: 23 Err: 885 (7.72%) Active: 768 Started: 768 Finished: 0
summary = 114933 in 00:05:21 = 358.4/s Avg: 0 Min: 0 Max: 74 Err: 8603 (7.49%)
summary + 11508 in 00:00:30 = 383.5/s Avg: 0 Min: 0 Max: 21 Err: 913 (7.93%) Active: 768 Started: 768 Finished: 0
summary = 126441 in 00:05:51 = 360.6/s Avg: 0 Min: 0 Max: 74 Err: 9516 (7.53%)
summary + 11498 in 00:00:30 = 383.4/s Avg: 0 Min: 0 Max: 21 Err: 896 (7.79%) Active: 768 Started: 768 Finished: 0
summary = 137939 in 00:06:21 = 362.4/s Avg: 0 Min: 0 Max: 74 Err: 10412 (7.55%)
summary + 11492 in 00:00:30 = 383.1/s Avg: 0 Min: 0 Max: 20 Err: 919 (8.00%) Active: 768 Started: 768 Finished: 0
summary = 149431 in 00:06:51 = 363.9/s Avg: 0 Min: 0 Max: 74 Err: 11331 (7.58%)
summary + 11533 in 00:00:30 = 384.4/s Avg: 0 Min: 0 Max: 26 Err: 887 (7.69%) Active: 768 Started: 768 Finished: 0
summary = 160964 in 00:07:21 = 365.3/s Avg: 0 Min: 0 Max: 74 Err: 12218 (7.59%)
summary + 11460 in 00:00:30 = 382.0/s Avg: 0 Min: 0 Max: 80 Err: 911 (7.95%) Active: 768 Started: 768 Finished: 0
summary = 172424 in 00:07:51 = 366.3/s Avg: 0 Min: 0 Max: 80 Err: 13129 (7.61%)
summary + 11534 in 00:00:30 = 384.5/s Avg: 0 Min: 0 Max: 23 Err: 901 (7.81%) Active: 768 Started: 768 Finished: 0
summary = 183958 in 00:08:21 = 367.4/s Avg: 0 Min: 0 Max: 80 Err: 14030 (7.63%)
summary + 11514 in 00:00:30 = 383.8/s Avg: 0 Min: 0 Max: 23 Err: 909 (7.89%) Active: 768 Started: 768 Finished: 0
summary = 195472 in 00:08:51 = 368.4/s Avg: 0 Min: 0 Max: 80 Err: 14939 (7.64%)
summary + 11488 in 00:00:30 = 383.0/s Avg: 0 Min: 0 Max: 10 Err: 904 (7.87%) Active: 768 Started: 768 Finished: 0
summary = 206960 in 00:09:21 = 369.1/s Avg: 0 Min: 0 Max: 80 Err: 15843 (7.66%)
summary + 11528 in 00:00:30 = 384.2/s Avg: 0 Min: 0 Max: 21 Err: 912 (7.91%) Active: 768 Started: 768 Finished: 0
summary = 218488 in 00:09:51 = 369.9/s Avg: 0 Min: 0 Max: 80 Err: 16755 (7.67%)

summary + 5648 in 00:00:15 = 378.3/s Avg: 0 Min: 0 Max: 20 Err: 353 (6.25%) Active: 0 Started: 768 Finished: 768
summary = 224136 in 00:10:06 = 370.1/s Avg: 0 Min: 0 Max: 80 Err: 17108 (7.63%)
Tidying up ... @ 2025 Mar 27 05:57:44 UTC (1743055064938)
... end of run

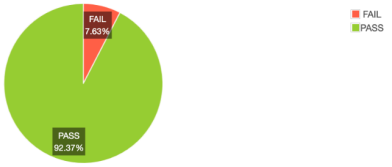
Test and Report information

Source file	"test-user-stress-v2-768-results.jtl"
Start Time	"3/27/25, 5:47 AM"
End Time	"3/27/25, 5:57 AM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.924	500 ms	1 sec 500 ms	Total
0.924	500 ms	1 sec 500 ms	HTTP Request

Requests Summary



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	224136	17108	7.63%	0.75	0	80	1.00	1.00	2.00	2.00	374.26	348.72	38.15
HTTP Request	224136	17108	7.63%	0.75	0	80	1.00	1.00	2.00	2.00	374.26	348.72	38.15

Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 172.31.1.46:80 failed to respond	17078	99.82%	7.62%
Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	30	0.18%	0.01%

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	224136	17108	Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 172.31.1.46:80 failed to respond	17078	Non HTTP response code: java.net.SocketException/ Non HTTP response message: Connection reset	30						
HTTP Request	224136	17108	Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 172.31.1.46:80 failed to respond	17078	Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset	30						

d. Analysis:
   i. **Reduced Error Rate**: The error rate decreased from ~25% (with 1000 users) to ~7.6% (with 768 users), which is a significant improvement but not error-free.
   ii. Error Analysis:

- **NoHttpResponseException (99.82% of errors)**: This overwhelmingly dominant error occurs when a TCP connection is established, but the server drops it before sending an HTTP response. This typically happens when a server accepts more connections than it can process.
- **SocketException: Connection reset (0.18% of errors)**: This much less common error occurs when the server actively closes a connection that was previously established.

4. 700 users x 500 requests per use (10 min)
    a. Configuration:

```
root@ip-172-31-15-56:~/test-0326# cat test-user-stress-v2-700.jmx
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
        <TestPlan guiclass="TestPlanGui" testclass="TestPlan"
testname="Enhanced User Stress Test Plan" enabled="true">
        <stringProp name="TestPlan.comments">Testing system
performance under 700 concurrent users with comprehensive
reporting</stringProp>
        <boolProp name="TestPlan.functional_mode">false</boolProp>
        <boolProp
name="TestPlan.serialize_threadgroups">false</boolProp>
        <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments" guiclass="ArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
        <stringProp name="Argument.name">HOST</stringProp>
        <stringProp name="Argument.value">172.31.1.46</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
        <stringProp name="Argument.name">PORT</stringProp>
        <stringProp name="Argument.value">80</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
```

```
        <stringProp name="Argument.name">PROTOCOL</stringProp>
        <stringProp name="Argument.value">http</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        </collectionProp>
        </elementProp>
        <stringProp name="TestPlan.user_define_classpath"></stringProp>
        </TestPlan>
        <hashTree>
        <ThreadGroup guiclass="ThreadGroupGui"
testclass="ThreadGroup" testname="User Stress Test Group"
enabled="true">
        <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp
name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
        </elementProp>
        <stringProp name="ThreadGroup.num_threads">700</stringProp>
        <stringProp name="ThreadGroup.ramp_time">30</stringProp>
        <boolProp name="ThreadGroup.scheduler">true</boolProp>
        <stringProp name="ThreadGroup.duration">600</stringProp>
        <stringProp name="ThreadGroup.delay">5</stringProp>
        </ThreadGroup>
        <hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="HTTP Request"
enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
```

<stringProp
name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
<stringProp name="HTTPSampler.contentEncoding"></stringProp>
<stringProp name="HTTPSampler.path">/</stringProp>
<stringProp name="HTTPSampler.method">GET</stringProp>
<boolProp name="HTTPSampler.follow_redirects">true</boolProp>
<boolProp name="HTTPSampler.auto_redirects">false</boolProp>
<boolProp name="HTTPSampler.use_keepalive">true</boolProp>
<boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
<stringProp
name="HTTPSampler.embedded_url_re"></stringProp>
<stringProp
name="HTTPSampler.connect_timeout">5000</stringProp>
<stringProp
name="HTTPSampler.response_timeout">30000</stringProp>
</HTTPSamplerProxy>
<hashTree>
<ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Response Code Assertion"
enabled="true">
<collectionProp name="Asserion.test_strings">
<stringProp name="49586">200</stringProp>
</collectionProp>
<stringProp name="Assertion.custom_message"></stringProp>
<stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
<boolProp name="Assertion.assume_success">false</boolProp>
<intProp name="Assertion.test_type">8</intProp>
</ResponseAssertion>
<hashTree/>
<DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time Assertion"
enabled="true">
<stringProp name="DurationAssertion.duration">1000</stringProp>
</DurationAssertion>
<hashTree/>
</hashTree>

```
        <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Realistic User Timing"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
        </UniformRandomTimer>
        <hashTree/>
        </hashTree>
        </hashTree>
  </hashTree>
</jmeterTestPlan>
```

b. Test:
   jmeter -n -t test-user-stress-v2-700.jmx -l test-user-stress-v2-700-results.jtl -e -o ./test-user-stress-v2-700-report

c. Result:
   ```
   root@ip-172-31-15-56:~/test-0326# jmeter -n -t
   test-user-stress-v2-700.jmx -l test-user-stress-v2-700-results.jtl -e -o
   ./test-user-stress-v2-700-report
   WARN StatusConsoleListener The use of package scanning to locate
   plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate
   plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate
   plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate
   plugins is deprecated and will be removed in a future release
   Creating summariser <summary>
   Created the tree successfully using test-user-stress-v2-700.jmx
   Starting standalone test @ 2025 Mar 27 06:09:18 UTC (1743055758722)
   Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
   message on port 4445
   summary +    143 in 00:00:11 =   13.2/s Avg:     3 Min:          0 Max:
   63 Err:          0 (0.00%) Active: 136 Started: 136 Finished: 0
   summary +   6815 in 00:00:30 =  227.2/s Avg:   0 Min:          0 Max:
   45 Err:          0 (0.00%) Active: 700 Started: 700 Finished: 0
   summary =   6958 in 00:00:41 =  170.2/s Avg:   1 Min:          0 Max:
   63 Err:          0 (0.00%)
   ```

summary + 10482 in 00:00:30 = 349.4/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 17440 in 00:01:11 = 246.1/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10506 in 00:00:30 = 350.2/s Avg: 0 Min: 0 Max: 28 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 27946 in 00:01:41 = 277.0/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10527 in 00:00:30 = 350.9/s Avg: 0 Min: 0 Max: 27 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 38473 in 00:02:11 = 294.0/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10561 in 00:00:30 = 352.1/s Avg: 0 Min: 0 Max: 30 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 49034 in 00:02:41 = 304.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10443 in 00:00:30 = 348.1/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 59477 in 00:03:11 = 311.6/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10470 in 00:00:30 = 349.0/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 69947 in 00:03:41 = 316.7/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10493 in 00:00:30 = 349.7/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 80440 in 00:04:11 = 320.6/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10495 in 00:00:30 = 349.9/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 90935 in 00:04:41 = 323.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10467 in 00:00:30 = 348.9/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 101402 in 00:05:11 = 326.2/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10488 in 00:00:30 = 349.5/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 111890 in 00:05:41 = 328.2/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)

summary + 10485 in 00:00:30 = 349.5/s Avg: 0 Min: 0 Max: 20 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 122375 in 00:06:11 = 330.0/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10510 in 00:00:30 = 350.3/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 132885 in 00:06:41 = 331.5/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10498 in 00:00:30 = 349.9/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 143383 in 00:07:11 = 332.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10545 in 00:00:30 = 351.6/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 153928 in 00:07:41 = 334.0/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10477 in 00:00:30 = 349.2/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 164405 in 00:08:11 = 334.9/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10488 in 00:00:30 = 349.7/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 174893 in 00:08:41 = 335.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10520 in 00:00:30 = 350.7/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 185413 in 00:09:11 = 336.6/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10517 in 00:00:30 = 350.5/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 195930 in 00:09:41 = 337.3/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 8568 in 00:00:25 = 347.3/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 0 Started: 700 Finished: 700
summary = 204498 in 00:10:06 = 337.7/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 27 06:19:24 UTC (1743056364673)
... end of run

d. Analysis

i. Total Requests: 204,498
Average Throughput: 337.7 requests per second
Error Rate: 0.00%
Max Response Time: 63ms

ii. **Our system's practical limit is approximately 700 concurrent users with the current configuration.**

5. Verification the limiting factor is Nginx's configuration of "worker_connection"
    a. Change configuration and restart Nginx:
    root@ip-172-31-1-46:~# cat /etc/nginx/nginx.conf | grep worker_connection
        worker_connections 1000;
    b. Test with 768 users x 500 requests per user (10 min)
    c. Result:
    root@ip-172-31-15-56:~/test-0326/verify_worker_connection# jmeter -n -t test-user-stress-v2-768.jmx -l test-user-stress-v2-768-results.jtl -e -o ./test-user-stress-v2-768-report
    WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
    Creating summariser <summary>
    Created the tree successfully using test-user-stress-v2-768.jmx
    Starting standalone test @ 2025 Mar 27 06:31:05 UTC (1743057065638)
    Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
    summary + 2095 in 00:00:24 = 87.4/s Avg: 1 Min: 0 Max: 71 Err: 0 (0.00%) Active: 486 Started: 486 Finished: 0
    summary + 10588 in 00:00:30 = 352.9/s Avg: 0 Min: 0 Max: 27 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
    summary = 12683 in 00:00:54 = 235.0/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
    summary + 11509 in 00:00:30 = 383.7/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0

summary = 24192 in 00:01:24 = 288.1/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11603 in 00:00:30 = 386.8/s Avg: 0 Min: 0 Max: 23 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 35795 in 00:01:54 = 314.1/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11469 in 00:00:30 = 382.4/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 47264 in 00:02:24 = 328.3/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11484 in 00:00:30 = 382.7/s Avg: 0 Min: 0 Max: 31 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 58748 in 00:02:54 = 337.7/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11534 in 00:00:30 = 384.5/s Avg: 0 Min: 0 Max: 20 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 70282 in 00:03:24 = 344.6/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11543 in 00:00:30 = 384.8/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 81825 in 00:03:54 = 349.7/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11515 in 00:00:30 = 383.8/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 93340 in 00:04:24 = 353.6/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11515 in 00:00:30 = 383.9/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 104855 in 00:04:54 = 356.7/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11521 in 00:00:30 = 384.0/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 116376 in 00:05:24 = 359.2/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11528 in 00:00:30 = 384.3/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 127904 in 00:05:54 = 361.3/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11516 in 00:00:30 = 383.9/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0

summary = 139420 in 00:06:24 = 363.1/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11534 in 00:00:30 = 384.2/s Avg: 0 Min: 0 Max: 23 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 150954 in 00:06:54 = 364.6/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11499 in 00:00:30 = 383.5/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 162453 in 00:07:24 = 365.9/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11573 in 00:00:30 = 385.8/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 174026 in 00:07:54 = 367.2/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11533 in 00:00:30 = 384.4/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 185559 in 00:08:24 = 368.2/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11477 in 00:00:30 = 382.6/s Avg: 0 Min: 0 Max: 24 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 197036 in 00:08:54 = 369.0/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11556 in 00:00:30 = 385.2/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 208592 in 00:09:24 = 369.9/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 11471 in 00:00:30 = 382.4/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 768 Started: 768 Finished: 0
summary = 220063 in 00:09:54 = 370.5/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
summary + 4372 in 00:00:12 = 376.7/s Avg: 0 Min: 0 Max: 20 Err: 0 (0.00%) Active: 0 Started: 768 Finished: 768
summary = 224435 in 00:10:06 = 370.6/s Avg: 0 Min: 0 Max: 71 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 27 06:41:11 UTC (1743057671611)
... end of run

d. Analysis:
This confirms that the worker_connections setting was indeed the primary bottleneck in your system. By increasing it from 768 to 1000,

we've successfully eliminated all the errors that were occurring in your previous 768-user test.

2 Throughput (Combined With Users) Stress Testing:

1. 100 RPS with 700 users
    a. Configuration:

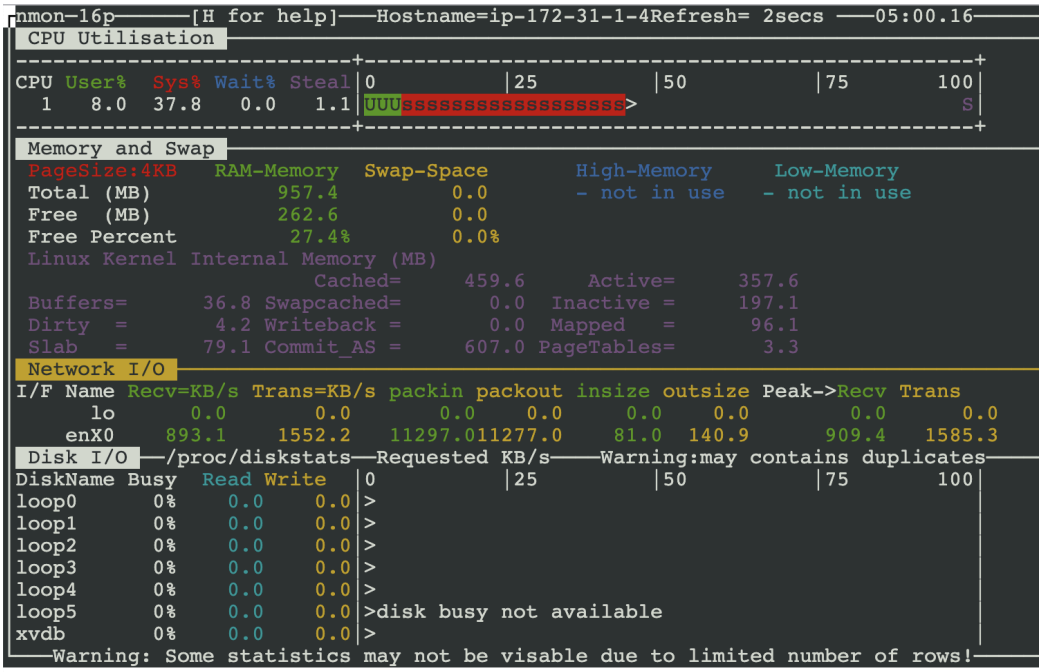
```
root@ip-172-31-15-56:~/test-0326# cat test-throughput-stress.jmx
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
        <TestPlan guiclass="TestPlanGui" testclass="TestPlan"
testname="Throughput Stress Test Plan" enabled="true">
        <stringProp name="TestPlan.comments">Testing system
throughput with 700 concurrent users</stringProp>
        <boolProp name="TestPlan.functional_mode">false</boolProp>
        <boolProp
name="TestPlan.serialize_threadgroups">false</boolProp>
        <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments" guiclass="ArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
        <stringProp name="Argument.name">HOST</stringProp>
        <stringProp name="Argument.value">172.31.1.46</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        </collectionProp>
        </elementProp>
        <stringProp name="TestPlan.user_define_classpath"></stringProp>
        </TestPlan>
        <hashTree>
        <ThreadGroup guiclass="ThreadGroupGui"
testclass="ThreadGroup" testname="Throughput Stress Test Group"
enabled="true">
        <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
```

```
        <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp
name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">-1</stringProp>
        </elementProp>
        <stringProp name="ThreadGroup.num_threads">700</stringProp>
        <stringProp name="ThreadGroup.ramp_time">30</stringProp>
        <boolProp name="ThreadGroup.scheduler">true</boolProp>
        <stringProp name="ThreadGroup.duration">600</stringProp>
        <stringProp name="ThreadGroup.delay">5</stringProp>
        </ThreadGroup>
        <hashTree>
        <ConstantThroughputTimer
guiclass="ConstantThroughputTimerGui"
testclass="ConstantThroughputTimer" testname="Throughput Timer"
enabled="true">
        <stringProp
name="ConstantThroughputTimer.throughput">6000</stringProp>
        <stringProp
name="ConstantThroughputTimer.calcMode">2</stringProp>
        </ConstantThroughputTimer>
        <hashTree/>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="HTTP Request"
enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments" guiclass="HTTPArgumentsPanel"
testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">80</stringProp>
        <stringProp name="HTTPSampler.protocol">http</stringProp>
        <stringProp name="HTTPSampler.path">/</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <stringProp
name="HTTPSampler.connect_timeout">5000</stringProp>
```

```
        <stringProp
name="HTTPSampler.response_timeout">30000</stringProp>
        </HTTPSamplerProxy>
        <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Response Code Assertion"
enabled="true">
        <collectionProp name="Asserion.test_strings">
        <stringProp name="49586">200</stringProp>
        </collectionProp>
        <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        </hashTree>
        </hashTree>
        </hashTree>
  </hashTree>
</jmeterTestPlan>
```

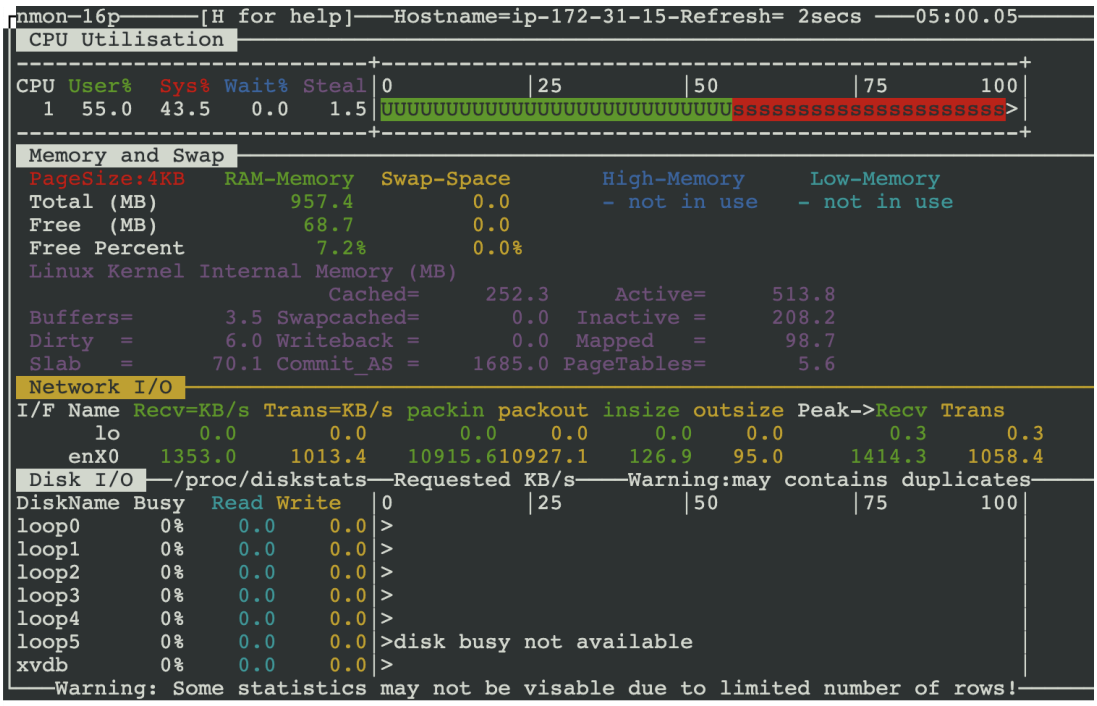
b. Test:
   jmeter -n -t test-throughput-stress.jmx -l test-throughput-100RPS-results.jtl
   -e -o ./test-throughput-100RPS-report

c. Resources monitoring:



i-09eac4388dcf36619 (Nginx Server1)

PublicIPs: 52.53.164.74 PrivateIPs: 172.31.1.46



i-0fb1802dc4d3c8147 (JMeter)

PublicIPs: 54.219.134.79 PrivateIPs: 172.31.15.56

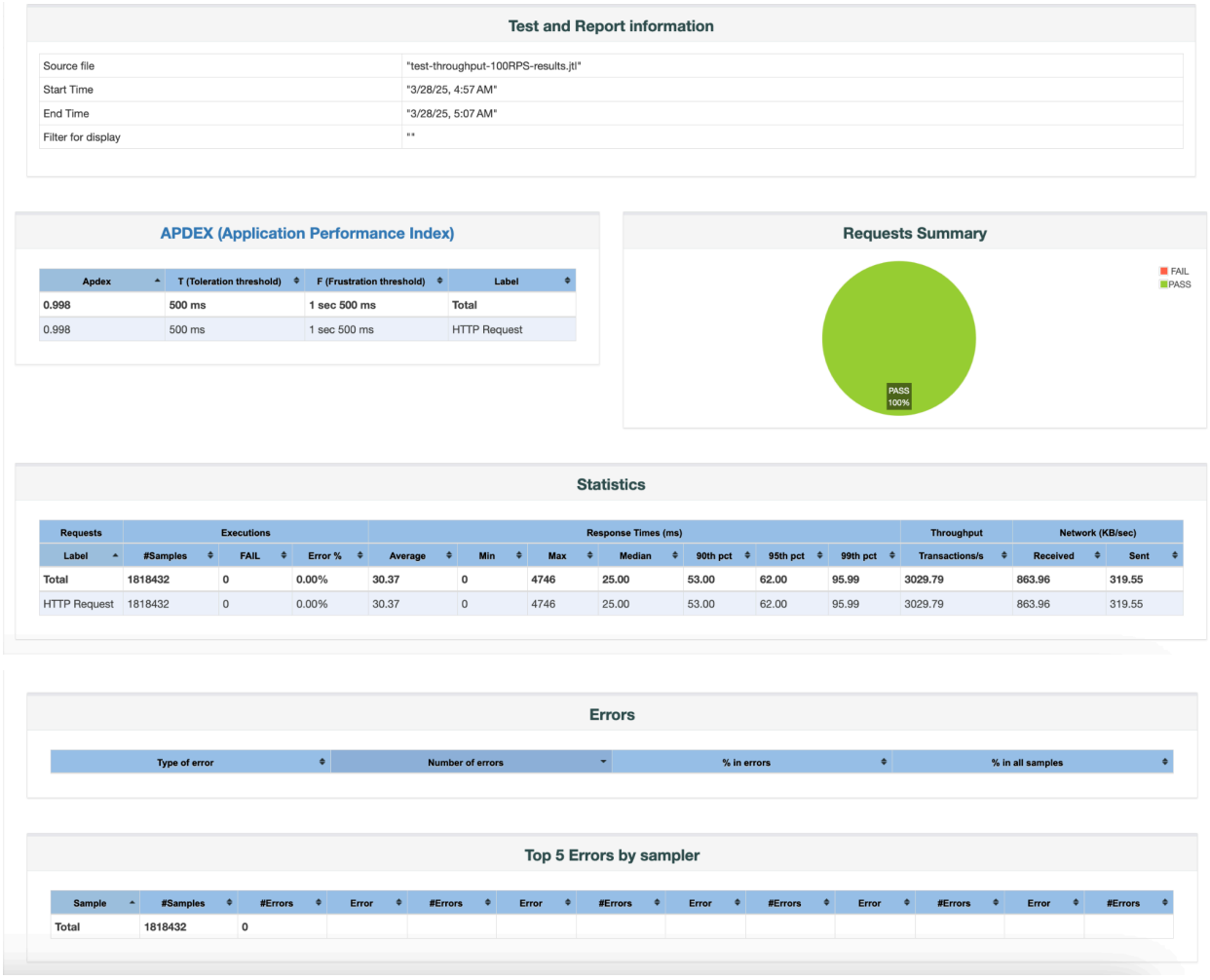
On the JMeter instance, CPU usage is almost full.

d. Result:

```
root@ip-172-31-15-56:~/test-0326# jmeter -n -t test-throughput-stress.jmx
-l test-throughput-100RPS-results.jtl -e -o ./test-throughput-100RPS-report
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-throughput-stress.jmx
Starting standalone test @ 2025 Mar 28 04:57:33 UTC (1743137853196)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
message on port 4445
summary +  32956 in 00:00:26 = 1249.0/s Avg: 34 Min:         1 Max:   870
Err:     0 (0.00%) Active: 499 Started: 499 Finished: 0
summary +  72593 in 00:00:30 = 2420.0/s Avg: 55 Min:         0 Max:  4746
Err:     0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 105549 in 00:00:56 = 1871.9/s Avg:         48 Min:         0
Max:  4746 Err:       0 (0.00%)
summary +  75888 in 00:00:30 = 2529.6/s Avg: 45 Min:         0 Max:  3410
Err:     0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 181437 in 00:01:26 = 2100.3/s Avg:         47 Min:         0
Max:  4746 Err:       0 (0.00%)
summary +  79425 in 00:00:30 = 2646.5/s Avg: 35 Min:         0 Max:  1860
Err:     0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 260862 in 00:01:56 = 2241.1/s Avg:         43 Min:         0
Max:  4746 Err:       0 (0.00%)
summary +  82583 in 00:00:30 = 2753.9/s Avg: 34 Min:         0 Max:  2396
Err:     0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 343445 in 00:02:26 = 2346.2/s Avg:         41 Min:         0
Max:  4746 Err:       0 (0.00%)
summary +  84963 in 00:00:30 = 2832.1/s Avg: 27 Min:         0 Max:  1256
Err:     0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 428408 in 00:02:56 = 2428.8/s Avg:         38 Min:         0
Max:  4746 Err:       0 (0.00%)
```

summary + 86118 in 00:00:30 = 2870.6/s Avg: 28 Min: 0 Max: 2099 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 514526 in 00:03:26 = 2492.9/s Avg: 37 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 89038 in 00:00:30 = 2969.1/s Avg: 27 Min: 0 Max: 1046 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 603564 in 00:03:56 = 2553.3/s Avg: 35 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 90980 in 00:00:30 = 3032.7/s Avg: 27 Min: 0 Max: 955 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 694544 in 00:04:26 = 2607.2/s Avg: 34 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 92033 in 00:00:30 = 3069.0/s Avg: 29 Min: 0 Max: 878 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 786577 in 00:04:56 = 2653.8/s Avg: 34 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 93716 in 00:00:30 = 3125.2/s Avg: 32 Min: 0 Max: 1612 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 880293 in 00:05:26 = 2697.1/s Avg: 33 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 95668 in 00:00:30 = 3188.9/s Avg: 27 Min: 0 Max: 700 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 975961 in 00:05:56 = 2738.5/s Avg: 33 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 97536 in 00:00:30 = 3251.2/s Avg: 27 Min: 0 Max: 897 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1073497 in 00:06:26 = 2778.3/s Avg: 32 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 100241 in 00:00:30 = 3341.4/s Avg: 26 Min: 0 Max: 561 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1173738 in 00:06:56 = 2818.8/s Avg: 32 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 102207 in 00:00:30 = 3407.8/s Avg: 22 Min: 0 Max: 638 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1275945 in 00:07:26 = 2858.4/s Avg: 31 Min: 0 Max: 4746 Err: 0 (0.00%)
summary + 100972 in 00:00:30 = 3366.3/s Avg: 27 Min: 0 Max: 1248 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1376917 in 00:07:56 = 2890.3/s Avg: 31 Min: 0 Max: 4746 Err: 0 (0.00%)

```
summary + 101467 in 00:00:30 = 3382.2/s Avg:          31 Min:          0
Max:  1353 Err:        0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1478384 in 00:08:26 = 2919.5/s Avg:         31 Min:          0
Max:  4746 Err:        0 (0.00%)
summary + 101568 in 00:00:30 = 3385.0/s Avg:          30 Min:          0
Max:  1623 Err:        0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1579952 in 00:08:56 = 2945.5/s Avg:         31 Min:          0
Max:  4746 Err:        0 (0.00%)
summary + 102605 in 00:00:30 = 3420.7/s Avg:          25 Min:          0
Max:   662 Err:        0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1682557 in 00:09:26 = 2970.7/s Avg:         30 Min:          0
Max:  4746 Err:        0 (0.00%)
summary + 102694 in 00:00:30 = 3423.1/s Avg:          24 Min:          0
Max:   851 Err:        0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1785251 in 00:09:56 = 2993.5/s Avg:         30 Min:          0
Max:  4746 Err:        0 (0.00%)
summary +  33181 in 00:00:09 = 3517.9/s Avg: 27 Min:          0 Max:   597
Err:     0 (0.00%) Active: 0 Started: 700 Finished: 700
summary = 1818432 in 00:10:06 = 3001.6/s Avg:         30 Min:          0
Max:  4746 Err:        0 (0.00%)
Tidying up ... @ 2025 Mar 28 05:07:39 UTC (1743138459434)
... end of run
```



Test and Report information

Source file	"test-throughput-100RPS-results.jtl"
Start Time	"3/28/25, 4:57 AM"
End Time	"3/28/25, 5:07 AM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.998	500 ms	1 sec 500 ms	Total
0.998	500 ms	1 sec 500 ms	HTTP Request

Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	1818432	0	0.00%	30.37	0	4746	25.00	53.00	62.00	95.99	3029.79	863.96	319.55
HTTP Request	1818432	0	0.00%	30.37	0	4746	25.00	53.00	62.00	95.99	3029.79	863.96	319.55

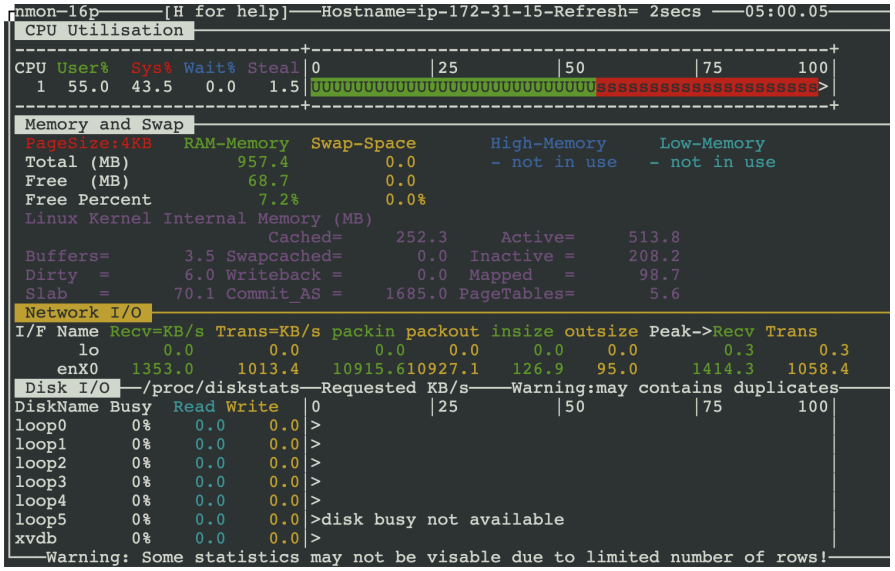
Errors

Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	1818432	0										

e. Analysis:
   i. **Test Duration**: 10 minutes 6 seconds
      Total Requests: 1,818,432
      Average Throughput: 3,001.6 requests per second
      Error Rate: 0% (no errors)
      Concurrent Users: 700
   ii. We targeted 100 RPS (6000 requests per minute) but achieved 3,001.6 RPS
      1. Potential cause of the discrepancy: JMeter Resource Limitations
         a. During the test, JMeter's CPU usage is almost 100%



b. JMeter itself may not have enough CPU resources to accurately enforce the timer constraints while managing 700 active threads.
c. JMeter prioritizes executing test logic over enforcing timing constraints. Essentially, the test became a "as fast as possible" test rather than a controlled throughput test.

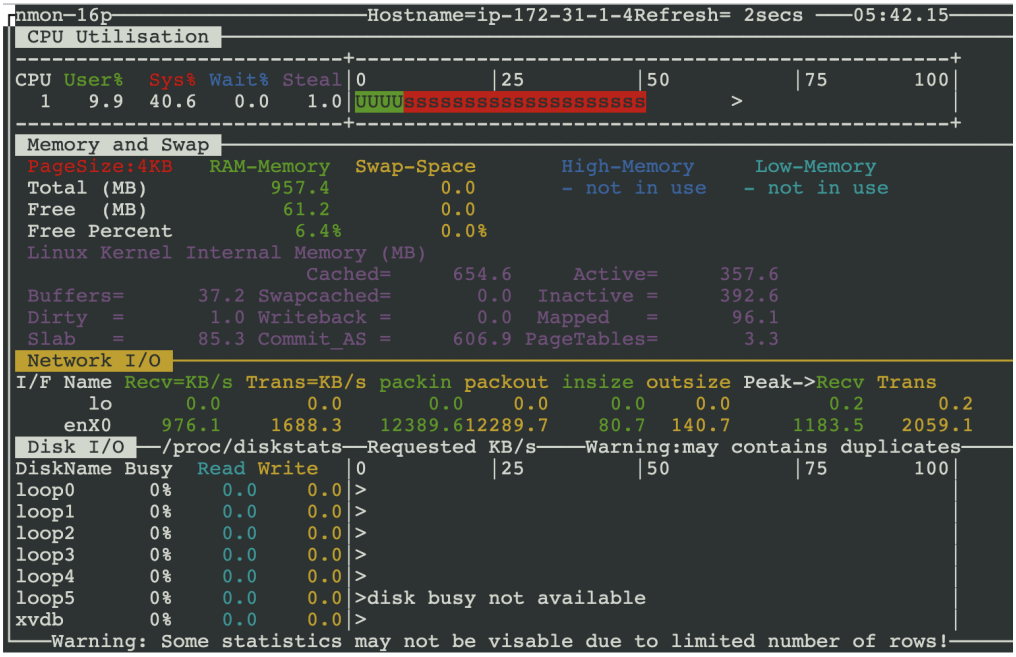
iii. Since the test became a “as fast as possible” test rather than a controlled throughput test, it shows:

1. The Maximum System Throughput With 700 Users: The system was able to handle approximately 3,000 RPS (requests per second) with 700 concurrent users without any errors.
2. Performance Under Load: Average response time of 30ms. Most requests were completed very quickly.
3. System Stability: The complete absence of errors despite the high load indicates the system is very stable.
4. The test reached the limits of what the JMeter instance can generate rather than what the Nginx system can handle. The actual system capacity may be even higher.

2. 10 RPS with 700 users
   a. Test plan changes change:

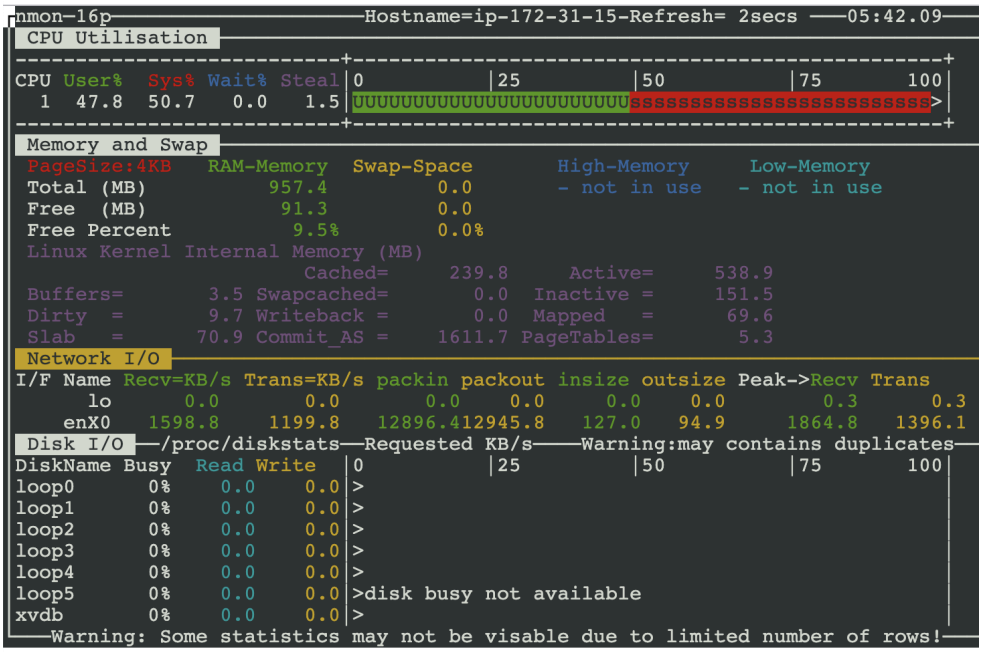
<stringProp
name="ConstantThroughputTimer.throughput">600</stringProp>

b. Resource monitoring



i-09eac4388dcf36619 (Nginx Server1)

PublicIPs: 52.53.164.74 PrivateIPs: 172.31.1.46



On the JMeter instance, CPU usage is still almost full.

c. Result:

root@ip-172-31-15-56:~/test-0326# jmeter -n -t test-throughput-stress2.jmx -l test-throughput-10RPS-results.jtl -e -o ./test-throughput-10RPS-report
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-throughput-stress2.jmx
Starting standalone test @ 2025 Mar 28 05:39:58 UTC (1743140398268)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 1 in 00:00:06 = 0.2/s Avg: 89 Min: 89 Max: 89 Err: 0 (0.00%) Active: 12 Started: 12 Finished: 0

summary + 50970 in 00:00:26 = 1977.0/s Avg: 22 Min: 0 Max: 982 Err: 0 (0.00%) Active: 616 Started: 616 Finished: 0
summary = 50971 in 00:00:31 = 1627.1/s Avg: 22 Min: 0 Max: 982 Err: 0 (0.00%)
summary + 78442 in 00:00:30 = 2615.0/s Avg: 29 Min: 0 Max: 1000 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 129413 in 00:01:01 = 2109.5/s Avg: 26 Min: 0 Max: 1000 Err: 0 (0.00%)
summary + 83245 in 00:00:30 = 2777.1/s Avg: 27 Min: 0 Max: 1023 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 212658 in 00:01:31 = 2328.6/s Avg: 26 Min: 0 Max: 1023 Err: 0 (0.00%)
summary + 88035 in 00:00:30 = 2934.5/s Avg: 32 Min: 0 Max: 1483 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 300693 in 00:02:01 = 2477.7/s Avg: 28 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 92726 in 00:00:30 = 3093.1/s Avg: 22 Min: 0 Max: 548 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 393419 in 00:02:31 = 2599.6/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 94764 in 00:00:30 = 3160.4/s Avg: 25 Min: 0 Max: 1360 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 488183 in 00:03:01 = 2692.2/s Avg: 26 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 96810 in 00:00:30 = 3225.1/s Avg: 30 Min: 0 Max: 1307 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 584993 in 00:03:31 = 2767.9/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 98496 in 00:00:30 = 3286.3/s Avg: 30 Min: 0 Max: 703 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 683489 in 00:04:01 = 2832.2/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 97544 in 00:00:30 = 3251.5/s Avg: 27 Min: 0 Max: 795 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 781033 in 00:04:31 = 2878.4/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 98117 in 00:00:30 = 3272.7/s Avg: 26 Min: 0 Max: 997 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 879150 in 00:05:01 = 2917.6/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)

summary + 97631 in 00:00:30 = 3253.2/s Avg: 27 Min: 0 Max: 677 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 976781 in 00:05:31 = 2948.0/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 98794 in 00:00:30 = 3294.5/s Avg: 27 Min: 0 Max: 819 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1075575 in 00:06:01 = 2976.8/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 98971 in 00:00:30 = 3299.0/s Avg: 26 Min: 0 Max: 632 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1174546 in 00:06:31 = 3001.5/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 98839 in 00:00:30 = 3294.6/s Avg: 26 Min: 0 Max: 756 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1273385 in 00:07:01 = 3022.3/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 96928 in 00:00:30 = 3230.9/s Avg: 31 Min: 0 Max: 932 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1370313 in 00:07:31 = 3036.2/s Avg: 27 Min: 0 Max: 1483 Err: 0 (0.00%)
summary + 98286 in 00:00:30 = 3276.2/s Avg: 33 Min: 0 Max: 1616 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1468599 in 00:08:01 = 3050.8/s Avg: 28 Min: 0 Max: 1616 Err: 0 (0.00%)
summary + 99300 in 00:00:30 = 3316.4/s Avg: 31 Min: 0 Max: 1202 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1567899 in 00:08:31 = 3066.3/s Avg: 28 Min: 0 Max: 1616 Err: 0 (0.00%)
summary + 98541 in 00:00:30 = 3284.8/s Avg: 23 Min: 0 Max: 539 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1666440 in 00:09:01 = 3078.4/s Avg: 28 Min: 0 Max: 1616 Err: 0 (0.00%)
summary + 98445 in 00:00:30 = 3283.1/s Avg: 31 Min: 0 Max: 858 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1764885 in 00:09:31 = 3089.1/s Avg: 28 Min: 0 Max: 1616 Err: 0 (0.00%)
summary + 99035 in 00:00:30 = 3301.1/s Avg: 30 Min: 0 Max: 600 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1863920 in 00:10:01 = 3099.7/s Avg: 28 Min: 0 Max: 1616 Err: 0 (0.00%)

summary + 13549 in 00:00:04 = 3138.5/s Avg: 27 Min: 0 Max: 416 Err: 0 (0.00%) Active: 0 Started: 700 Finished: 700
summary = 1877469 in 00:10:06 = 3099.9/s Avg: 28 Min: 0 Max: 1616 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 28 05:50:04 UTC (1743141004327)
... end of run

d. Analysis:
    i. Still, we targeted 10 RPS (600 requests per minute) but achieved 3,001.6 RPS. It still indicates that JMeter itself may not have enough CPU resources to accurately enforce the timer constraints even with 10 RPS and 700 threads.

3. 1000 RPS with 700 users

a. Test plan changes:
<stringProp name="ConstantThroughputTimer.throughput">60000</stringProp>

b. Result:
root@ip-172-31-15-56:~/test-0326# jmeter -n -t test-throughput-stress3.jmx -l test-throughput-1000RPS-results.jtl -e -o ./test-throughput-1000RPS-report
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-throughput-stress3.jmx
Starting standalone test @ 2025 Mar 28 05:54:04 UTC (1743141244833)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 31092 in 00:00:25 = 1256.4/s Avg: 40 Min: 1 Max: 1518 Err: 0 (0.00%) Active: 461 Started: 461 Finished: 0
summary + 73767 in 00:00:30 = 2459.4/s Avg: 34 Min: 0 Max: 2086 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 104859 in 00:00:55 = 1915.5/s Avg: 36 Min: 0 Max: 2086 Err: 0 (0.00%)
summary + 77841 in 00:00:30 = 2594.7/s Avg: 31 Min: 0 Max: 1851 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 182700 in 00:01:25 = 2156.0/s Avg: 34 Min: 0 Max: 2086 Err: 0 (0.00%)
summary + 82460 in 00:00:30 = 2748.7/s Avg: 29 Min: 1 Max: 1543 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 265160 in 00:01:55 = 2310.9/s Avg: 32 Min: 0 Max: 2086 Err: 0 (0.00%)
summary + 85819 in 00:00:30 = 2860.7/s Avg: 31 Min: 0 Max: 3211 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 350979 in 00:02:25 = 2424.8/s Avg: 32 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 88667 in 00:00:30 = 2956.0/s Avg: 28 Min: 0 Max: 1571 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 439646 in 00:02:55 = 2515.9/s Avg: 31 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 92241 in 00:00:30 = 3075.1/s Avg: 24 Min: 0 Max: 1462 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 531887 in 00:03:25 = 2597.9/s Avg: 30 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 94074 in 00:00:30 = 3135.8/s Avg: 24 Min: 0 Max: 1067 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 625961 in 00:03:55 = 2666.6/s Avg: 29 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 95061 in 00:00:30 = 3168.7/s Avg: 24 Min: 0 Max: 642 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 721022 in 00:04:25 = 2723.5/s Avg: 28 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 97609 in 00:00:30 = 3253.6/s Avg: 23 Min: 0 Max: 697 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 818631 in 00:04:55 = 2777.5/s Avg: 28 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 97960 in 00:00:30 = 3265.3/s Avg: 20 Min: 0 Max: 1282 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 916591 in 00:05:25 = 2822.5/s Avg: 27 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 102640 in 00:00:30 = 3422.2/s Avg: 20 Min: 0 Max: 606 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 1019231 in 00:05:55 = 2873.0/s Avg: 26 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 103600 in 00:00:30 = 3455.1/s Avg: 26 Min: 0 Max: 912 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1122831 in 00:06:25 = 2918.4/s Avg: 26 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 105629 in 00:00:30 = 3521.0/s Avg: 23 Min: 0 Max: 507 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1228460 in 00:06:55 = 2962.0/s Avg: 26 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 104315 in 00:00:30 = 3477.2/s Avg: 24 Min: 0 Max: 479 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1332775 in 00:07:25 = 2996.7/s Avg: 26 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 103850 in 00:00:30 = 3461.7/s Avg: 28 Min: 0 Max: 2181 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1436625 in 00:07:55 = 3026.1/s Avg: 26 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 106641 in 00:00:30 = 3554.7/s Avg: 26 Min: 0 Max: 991 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1543266 in 00:08:25 = 3057.5/s Avg: 26 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 107732 in 00:00:30 = 3591.1/s Avg: 24 Min: 0 Max: 1190 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1650998 in 00:08:55 = 3087.5/s Avg: 26 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 108097 in 00:00:30 = 3603.2/s Avg: 22 Min: 1 Max: 861 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1759095 in 00:09:25 = 3114.9/s Avg: 26 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 106639 in 00:00:30 = 3554.6/s Avg: 22 Min: 1 Max: 703 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 1865734 in 00:09:55 = 3137.1/s Avg: 25 Min: 0 Max: 3211 Err: 0 (0.00%)
summary + 38370 in 00:00:11 = 3468.6/s Avg: 23 Min: 0 Max: 571 Err: 0 (0.00%) Active: 0 Started: 700 Finished: 700
summary = 1904104 in 00:10:06 = 3143.1/s Avg: 25 Min: 0 Max: 3211 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 28 06:04:11 UTC (1743141851063)
... end of run

c. Analyze:
    i. Average throughput: 3,143.1 requests/second
    Error rate: 0% (excellent)
    Concurrent users: 700
    Average response time: 25 milliseconds (very good overall)

4. 10 RPS with 750 users

a. Test plan changes:
<stringProp name="ThreadGroup.num_threads">750</stringProp>
<stringProp
name="ConstantThroughputTimer.throughput">600</stringProp>

b. Results
root@ip-172-31-15-56:~/test-0326# jmeter -n -t
test-throughput-stress4.jmx -l test-throughput-750-10RPS-results.jtl -e -o
./test-throughput-750-10RPS-report
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-throughput-stress4.jmx
Starting standalone test @ 2025 Mar 28 06:07:09 UTC (1743142029559)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
message on port 4445
summary + 24233 in 00:00:20 = 1208.6/s Avg: 18 Min: 0 Max: 569
Err: 0 (0.00%) Active: 386 Started: 386 Finished: 0
summary + 73221 in 00:00:30 = 2439.1/s Avg: 30 Min: 0 Max: 1033
Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 97454 in 00:00:50 = 1946.3/s Avg: 27 Min: 0 Max: 1033
Err: 0 (0.00%)
summary + 80249 in 00:00:30 = 2676.8/s Avg: 32 Min: 0 Max: 1326
Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 177703 in 00:01:20 = 2219.9/s Avg: 29 Min: 0
Max: 1326 Err: 0 (0.00%)

summary + 85389 in 00:00:30 = 2846.3/s Avg: 33 Min: 0 Max: 2047 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 263092 in 00:01:50 = 2390.6/s Avg: 31 Min: 0 Max: 2047 Err: 0 (0.00%)
summary + 87628 in 00:00:30 = 2920.8/s Avg: 29 Min: 0 Max: 1246 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 350720 in 00:02:20 = 2504.2/s Avg: 30 Min: 0 Max: 2047 Err: 0 (0.00%)
summary + 92815 in 00:00:30 = 3093.8/s Avg: 22 Min: 0 Max: 519 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 443535 in 00:02:50 = 2607.7/s Avg: 29 Min: 0 Max: 2047 Err: 0 (0.00%)
summary + 95218 in 00:00:30 = 3177.5/s Avg: 27 Min: 0 Max: 1045 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 538753 in 00:03:20 = 2693.1/s Avg: 28 Min: 0 Max: 2047 Err: 0 (0.00%)
summary + 97683 in 00:00:30 = 3256.0/s Avg: 32 Min: 0 Max: 2901 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 636436 in 00:03:50 = 2766.5/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 96870 in 00:00:30 = 3229.1/s Avg: 26 Min: 0 Max: 618 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 733306 in 00:04:20 = 2819.9/s Avg: 28 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 97947 in 00:00:30 = 3264.9/s Avg: 26 Min: 0 Max: 860 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 831253 in 00:04:50 = 2865.9/s Avg: 28 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 97755 in 00:00:30 = 3258.5/s Avg: 31 Min: 0 Max: 694 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 929008 in 00:05:20 = 2902.7/s Avg: 28 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 97022 in 00:00:30 = 3234.1/s Avg: 28 Min: 0 Max: 868 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1026030 in 00:05:50 = 2931.1/s Avg: 28 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 96710 in 00:00:30 = 3223.7/s Avg: 31 Min: 0 Max: 1267 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1122740 in 00:06:20 = 2954.2/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)

summary + 96158 in 00:00:30 = 3205.3/s Avg: 32 Min: 0 Max: 1166 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1218898 in 00:06:50 = 2972.6/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 97061 in 00:00:30 = 3235.4/s Avg: 28 Min: 0 Max: 1032 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1315959 in 00:07:20 = 2990.5/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 96301 in 00:00:30 = 3210.0/s Avg: 27 Min: 0 Max: 679 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1412260 in 00:07:50 = 3004.4/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 96199 in 00:00:30 = 3205.5/s Avg: 28 Min: 0 Max: 849 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1508459 in 00:08:20 = 3016.5/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 97212 in 00:00:30 = 3242.6/s Avg: 29 Min: 0 Max: 910 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1605671 in 00:08:50 = 3029.3/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 97903 in 00:00:30 = 3263.4/s Avg: 28 Min: 0 Max: 729 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1703574 in 00:09:20 = 3041.8/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 87543 in 00:00:30 = 2918.1/s Avg: 36 Min: 0 Max: 1147 Err: 0 (0.00%) Active: 750 Started: 750 Finished: 0
summary = 1791117 in 00:09:50 = 3035.5/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
summary + 50833 in 00:00:16 = 3239.4/s Avg: 29 Min: 0 Max: 1198 Err: 0 (0.00%) Active: 0 Started: 750 Finished: 750
summary = 1841950 in 00:10:06 = 3040.8/s Avg: 29 Min: 0 Max: 2901 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 28 06:17:15 UTC (1743142635692)
... end of run

c. Analyze:
    i. Average throughput: 3,040.8 requests/second
    Error rate: 0% (excellent)
    Concurrent users: 750
    Average response time: 29 milliseconds (very good)

5. 10 RPS with 1000 users
    a. Test plan changes:
    <stringProp name="ThreadGroup.num_threads">1000</stringProp>
    <stringProp
    name="ConstantThroughputTimer.throughput">600</stringProp>

    b. Results:
    root@ip-172-31-15-56:~/test-0326# jmeter -n -t
    test-throughput-stress5.jmx -l test-throughput-1000-10RPS-results.jtl -e -o
    ./test-throughput-1000-10RPS-repo
    rt
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    Creating summariser <summary>
    Created the tree successfully using test-throughput-stress5.jmx
    Starting standalone test @ 2025 Mar 28 06:23:13 UTC (1743142993818)
    Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
    message on port 4445
    summary + 16327 in 00:00:16 = 1031.0/s Avg: 17 Min: 0 Max: 990
    Err: 0 (0.00%) Active: 361 Started: 361 Finished: 0
    summary + 69633 in 00:00:30 = 2321.1/s Avg: 27 Min: 0 Max: 933
    Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
    summary = 85960 in 00:00:46 = 1875.3/s Avg: 25 Min: 0 Max: 990
    Err: 0 (0.00%)
    summary + 73826 in 00:00:30 = 2460.9/s Avg: 33 Min: 0 Max: 1553
    Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
    summary = 159786 in 00:01:16 = 2106.5/s Avg: 29 Min: 0
    Max: 1553 Err: 0 (0.00%)
    summary + 77820 in 00:00:30 = 2595.5/s Avg: 33 Min: 0 Max: 1668
    Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
    summary = 237606 in 00:01:46 = 2244.9/s Avg: 30 Min: 0
    Max: 1668 Err: 0 (0.00%)
    summary + 80880 in 00:00:30 = 2696.4/s Avg: 28 Min: 0 Max: 793
    Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0

summary = 318486 in 00:02:16 = 2344.6/s Avg: 30 Min: 0 Max: 1668 Err: 0 (0.00%)
summary + 86072 in 00:00:30 = 2869.1/s Avg: 35 Min: 0 Max: 1413 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 404558 in 00:02:46 = 2439.5/s Avg: 31 Min: 0 Max: 1668 Err: 0 (0.00%)
summary + 92660 in 00:00:30 = 3088.7/s Avg: 26 Min: 0 Max: 648 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 497218 in 00:03:16 = 2539.0/s Avg: 30 Min: 0 Max: 1668 Err: 0 (0.00%)
summary + 94446 in 00:00:30 = 3148.2/s Avg: 30 Min: 0 Max: 1463 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 591664 in 00:03:46 = 2619.9/s Avg: 30 Min: 0 Max: 1668 Err: 0 (0.00%)
summary + 95943 in 00:00:30 = 3198.1/s Avg: 31 Min: 0 Max: 1543 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 687607 in 00:04:16 = 2687.7/s Avg: 30 Min: 0 Max: 1668 Err: 0 (0.00%)
summary + 98840 in 00:00:30 = 3294.9/s Avg: 33 Min: 0 Max: 1786 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 786447 in 00:04:46 = 2751.4/s Avg: 30 Min: 0 Max: 1786 Err: 0 (0.00%)
summary + 99203 in 00:00:30 = 3306.8/s Avg: 27 Min: 0 Max: 1412 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 885650 in 00:05:16 = 2804.1/s Avg: 30 Min: 0 Max: 1786 Err: 0 (0.00%)
summary + 97529 in 00:00:30 = 3251.0/s Avg: 34 Min: 0 Max: 2764 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 983179 in 00:05:46 = 2842.9/s Avg: 30 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 97608 in 00:00:30 = 3253.6/s Avg: 22 Min: 0 Max: 434 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 1080787 in 00:06:16 = 2875.7/s Avg: 30 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 96923 in 00:00:30 = 3230.7/s Avg: 31 Min: 0 Max: 951 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 1177710 in 00:06:46 = 2901.9/s Avg: 30 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 96264 in 00:00:30 = 3208.9/s Avg: 25 Min: 0 Max: 809 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0

summary = 1273974 in 00:07:16 = 2923.1/s Avg: 29 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 96470 in 00:00:30 = 3215.7/s Avg: 26 Min: 0 Max: 657 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 1370444 in 00:07:46 = 2941.8/s Avg: 29 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 96858 in 00:00:30 = 3230.8/s Avg: 29 Min: 0 Max: 1104 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 1467302 in 00:08:16 = 2959.2/s Avg: 29 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 98191 in 00:00:30 = 3273.0/s Avg: 27 Min: 0 Max: 851 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 1565493 in 00:08:46 = 2977.2/s Avg: 29 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 95996 in 00:00:30 = 3199.9/s Avg: 23 Min: 0 Max: 674 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 1661489 in 00:09:16 = 2989.2/s Avg: 29 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 95244 in 00:00:30 = 3174.8/s Avg: 26 Min: 0 Max: 1676 Err: 0 (0.00%) Active: 1000 Started: 1000 Finished: 0
summary = 1756733 in 00:09:46 = 2998.7/s Avg: 29 Min: 0 Max: 2764 Err: 0 (0.00%)
summary + 64000 in 00:00:20 = 3185.7/s Avg: 34 Min: 0 Max: 2082 Err: 0 (0.00%) Active: 0 Started: 1000 Finished: 1000
summary = 1820733 in 00:10:06 = 3004.9/s Avg: 29 Min: 0 Max: 2764 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 28 06:33:20 UTC (1743143600092)
Error generating the report:
org.apache.jmeter.report.dashboard.GenerationException: Error while processing samples: Consumer failed with message :Consumer failed with message :Consumer failed with message :Consumer failed with message :Consumer failed with message :Mismatch between expected number of columns:17 and columns in CSV file:13, check your jmeter.save.saveservice.* configuration or check if line 1023060 in 'temp/latencyVsRequest/latencyVsRequest2135721706226370539-0' is complete
... end of run

c. Analysis:
    i. Average throughput: 3,004.9 requests/second
       Error rate: 0%

Concurrent users: 1,000
Average response time: 29 milliseconds

6. 10 RPS with 3000 users
    a. Test plan changes:
    <stringProp name="ThreadGroup.num_threads">3000</stringProp>
    <stringProp
    name="ConstantThroughputTimer.throughput">600</stringProp>

    b. Test:
    root@ip-172-31-15-56:~/test-0326# jmeter -n -t
    test-throughput-stress6.jmx -l test-throughput-3000-10RPS-results.jtl -e -o
    ./test-throughput-3000-10RPS-repo
    rt
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    Creating summariser <summary>
    Created the tree successfully using test-throughput-stress6.jmx
    Starting standalone test @ 2025 Mar 28 06:40:50 UTC (1743144050959)
    Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
    message on port 4445
    [5.150s][warning][os,thread] Failed to start thread "Unknown thread" -
    pthread_create failed (EAGAIN) for attributes: stacksize: 1024k,
    guardsize: 0k, detached.
    [5.150s][warning][os,thread] Failed to start the native thread for
    java.lang.Thread "Throughput Stress Test Group 1-2455"
    summary +   2292 in 00:00:09 = 265.7/s Avg: 35 Min: 3 Max: 243
    Err: 0 (0.00%) Active: 360 Started: 360 Finished: 0
    Killed
    c. Analysis:
        i. Critical warning: `Failed to start thread "Unknown thread" - pthread_create failed (EAGAIN)`

The EAGAIN error suggests resource exhaustion - likely the system ran out of memory or hit OS thread limits

7. 10 RPS with 2000 threads
    a. Test plan changes:
    <stringProp name="ThreadGroup.num_threads">2000</stringProp>
    <stringProp name="ConstantThroughputTimer.throughput">600</stringProp>

    b. Test:
    root@ip-172-31-15-56:~/test-0326# jmeter -n -t test-throughput-stress6.jmx -l test-throughput-2000-10RPS-results.jtl -e -o ./test-throughput-2000-10RPS-repo
    rt
    WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
    Creating summariser <summary>
    Created the tree successfully using test-throughput-stress6.jmx
    Starting standalone test @ 2025 Mar 28 06:42:32 UTC (1743144152395)
    Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
    summary + 30642 in 00:00:27 = 1126.8/s Avg: 28 Min: 0 Max: 740 Err: 0 (0.00%) Active: 1501 Started: 1501 Finished: 0
    (Then the host got stuck)
    c. Analysis:
        i. The system "got stuck" after processing just over 1,500 concurrent users. While no specific error message appears in the log, the test didn't proceed normally
        ii. Like the 3,000-user test, this test likely encountered resource constraints, though less catastrophically. This test further narrows down our system's concurrent user limit to approximately 1,500 users.

8. 10 RPS with 1500 users
    a. Test plan changes:
    <stringProp name="ThreadGroup.num_threads">1500</stringProp>
    <stringProp
    name="ConstantThroughputTimer.throughput">600</stringProp>

    b. Test:
    root@ip-172-31-15-56:~/test-0326# jmeter -n -t
    test-throughput-stress6.jmx -l test-throughput-1500-10RPS-results.jtl -e -o
    ./test-throughput-1500-10RPS-repo
    rt
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    WARN StatusConsoleListener The use of package scanning to locate
    plugins is deprecated and will be removed in a future release
    Creating summariser <summary>
    Created the tree successfully using test-throughput-stress6.jmx
    Starting standalone test @ 2025 Mar 28 07:01:34 UTC (1743145294026)
    Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
    message on port 4445
    summary + 31261 in 00:00:26 = 1223.4/s Avg: 28 Min: 0 Max: 911
    Err: 0 (0.00%) Active: 1032 Started: 1032 Finished: 0
    summary + 66562 in 00:00:30 = 2218.7/s Avg: 29 Min: 0 Max: 1279
    Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
    summary = 97823 in 00:00:56 = 1760.9/s Avg: 29 Min: 0 Max: 1279
    Err: 0 (0.00%)
    summary + 70195 in 00:00:30 = 2339.8/s Avg: 24 Min: 0 Max: 894
    Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
    summary = 168018 in 00:01:26 = 1962.7/s Avg: 27 Min: 0
    Max: 1279 Err: 0 (0.00%)
    summary + 76767 in 00:00:30 = 2563.4/s Avg: 25 Min: 0 Max: 776
    Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
    summary = 244785 in 00:01:56 = 2118.4/s Avg:26 Min: 0 Max: 1279
    Err: 0 (0.00%)
    summary + 80212 in 00:00:30 = 2673.7/s Avg: 35 Min: 0 Max: 1245
    Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0

summary = 324997 in 00:02:26 = 2232.8/s Avg: 29 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 82545 in 00:00:30 = 2751.5/s Avg: 27 Min: 0 Max: 984 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 407542 in 00:02:56 = 2321.5/s Avg: 28 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 87239 in 00:00:30 = 2908.0/s Avg: 24 Min: 0 Max: 898 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 494781 in 00:03:26 = 2407.1/s Avg: 27 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 90751 in 00:00:30 = 3025.0/s Avg: 23 Min: 0 Max: 662 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 585532 in 00:03:56 = 2485.8/s Avg: 27 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 91832 in 00:00:30 = 3061.1/s Avg: 26 Min: 0 Max: 860 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 677364 in 00:04:26 = 2550.8/s Avg: 27 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 93349 in 00:00:30 = 3111.6/s Avg: 24 Min: 0 Max: 885 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 770713 in 00:04:56 = 2607.7/s Avg: 26 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 97758 in 00:00:30 = 3254.9/s Avg: 24 Min: 0 Max: 743 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 868471 in 00:05:26 = 2667.4/s Avg: 26 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 97523 in 00:00:30 = 3254.5/s Avg: 23 Min: 0 Max: 683 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 965994 in 00:05:56 = 2716.9/s Avg: 26 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 96397 in 00:00:30 = 3213.2/s Avg: 28 Min: 0 Max: 907 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 1062391 in 00:06:26 = 2755.5/s Avg: 26 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 96638 in 00:00:30 = 3221.3/s Avg: 23 Min: 0 Max: 501 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 1159029 in 00:06:56 = 2789.1/s Avg: 26 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 96942 in 00:00:30 = 3231.4/s Avg: 25 Min: 0 Max: 666 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0

summary = 1255971 in 00:07:26 = 2818.9/s Avg: 26 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 96724 in 00:00:30 = 3224.1/s Avg: 24 Min: 0 Max: 921 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 1352695 in 00:07:56 = 2844.3/s Avg: 25 Min: 0 Max: 1279 Err: 0 (0.00%)
summary + 87157 in 00:00:30 = 2908.7/s Avg: 35 Min: 0 Max: 1532 Err: 0 (0.00%) Active: 1500 Started: 1500 Finished: 0
summary = 1439852 in 00:08:26 = 2847.7/s Avg: 26 Min: 0 Max: 1532 Err: 0 (0.00%)
(here the instance got stuck)

c. Analysis:
   i. Throughput Growth: Started at ~1,223 RPS and eventually stabilized at ~3,200 to 3,250 RPS;
   Consistent Response Times: Average response times remained stable in the 23 - 35ms range;
   Reliability: Zero errors throughout the entire test;
   System Failure: After approximately 8.5 minutes, the system got stuck despite showing good performance metrics.
   ii. The system can handle 1500 concurrent users for short periods, but not sustainably. For long-running stability, staying below 1,000 concurrent users is safer

3 Type of Request Stress Testing:

1. Static html + 1000 user

a. Create a Test HTML File

```
root@ip-172-31-1-46:~# sudo mkdir -p /var/www/html
echo "<html><body><h1>JMeter Test Page</h1></body></html>" | sudo tee /var/www/html
/index.html
<html><body><h1>JMeter Test Page</h1></body></html>
```

b. Test the html file works

```
root@ip-172-31-1-46:~# curl http://172.31.1.46/index.html
<html><body><h1>JMeter Test Page</h1></body></html>
```

c. Configuration

oot@ip-172-31-15-56:~/test-0326/user-stress-tests# cat test-static-stress-1000.jmx

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
```

```
  <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Static
HTML Stress Test Plan" enabled="true">
    <stringProp name="TestPlan.comments">Stress test static HTML with 1000
concurrent users</stringProp>
    <boolProp name="TestPlan.functional_mode">false</boolProp>
    <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
    <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
      <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
          <stringProp name="Argument.name">HOST</stringProp>
          <stringProp name="Argument.value">172.31.1.46</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
          <stringProp name="Argument.name">PORT</stringProp>
          <stringProp name="Argument.value">80</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
          <stringProp name="Argument.name">PROTOCOL</stringProp>
          <stringProp name="Argument.value">http</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
      </collectionProp>
    </elementProp>
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
  <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Static HTML Stress Group" enabled="true">
      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">1000</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
```

```
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /index.html" enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/index.html</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp name="HTTPSampler.response_timeout">30000</stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time &lt; 1000ms"
enabled="true">
          <stringProp name="DurationAssertion.duration">1000</stringProp>
        </DurationAssertion>
        <hashTree/>
      </hashTree>
      <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="User Timing Delay"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
      </UniformRandomTimer>
```

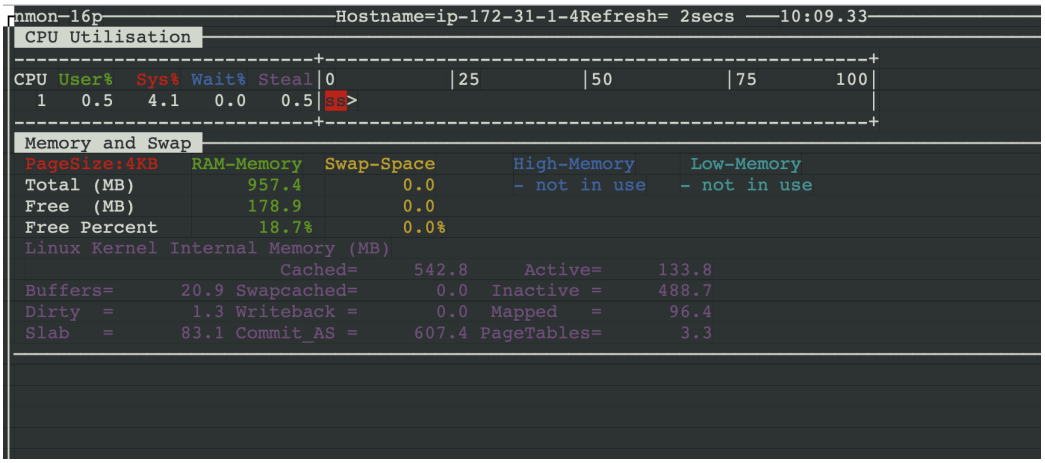
```
        <hashTree/>
      </hashTree>
    </hashTree>
  </hashTree>
</jmeterTestPlan>
```

d. On Nginx, monitor CPU usage: nmon
e. Run test
   jmeter -n -t test-static-stress-1000.jmx -l test-static-stress-1000.jtl -e -o ./test-static-stress-1000-report

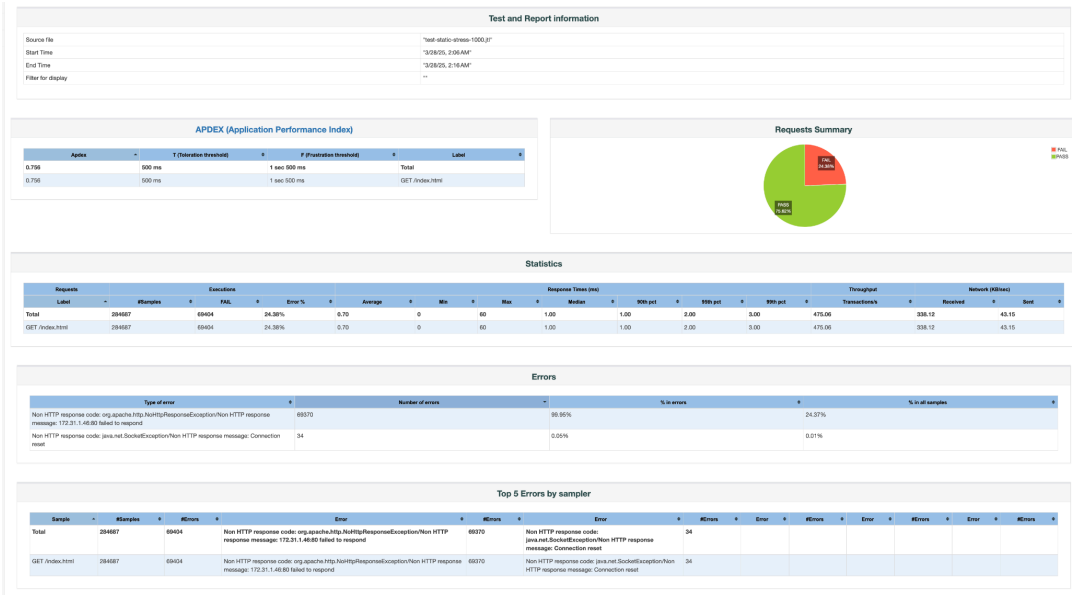
f. Test Result
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   Creating summariser <summary>
   Created the tree successfully using test-static-stress-1000.jmx
   Starting standalone test @ 2025 Mar 28 10:04:30 UTC (1743156270364)
   Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
   summary + 2242 in 00:00:29 = 76.7/s Avg: 1 Min: 0 Max: 63 Err: 0 (0.00%) Active: 404 Started: 404 Finished: 0
   summary + 9606 in 00:00:30 = 320.3/s Avg: 0 Min: 0 Max: 30 Err: 472 (4.91%) Active: 904 Started: 904 Finished: 0
   summary = 11848 in 00:00:59 = 200.0/s Avg: 0 Min: 0 Max: 63 Err: 472 (3.98%)
   summary + 14792 in 00:00:30 = 493.1/s Avg: 0 Min: 0 Max: 41 Err: 3660 (24.74%) Active: 1000 Started: 1000 Finished: 0
   summary = 26640 in 00:01:29 = 298.5/s Avg: 0 Min: 0 Max: 63 Err: 4132 (15.51%)
   summary + 14952 in 00:00:30 = 498.4/s Avg: 0 Min: 0 Max: 29 Err: 3793 (25.37%) Active: 1000 Started: 1000 Finished: 0
   summary = 41592 in 00:01:59 = 348.8/s Avg: 0 Min: 0 Max: 63 Err: 7925 (19.05%)
   summary + 14969 in 00:00:30 = 499.0/s Avg: 0 Min: 0 Max: 28 Err: 3804 (25.41%) Active: 1000 Started: 1000 Finished: 0

summary = 56561 in 00:02:29 = 379.0/s Avg: 0 Min: 0 Max: 63 Err:
11729 (20.74%)
summary + 15029 in 00:00:30 = 500.9/s Avg: 0 Min: 0 Max: 27 Err: 3813
(25.37%) Active: 1000 Started: 1000 Finished: 0
summary = 71590 in 00:02:59 = 399.4/s Avg: 0 Min: 0 Max: 63 Err:
15542 (21.71%)
summary + 15000 in 00:00:30 = 500.0/s Avg: 0 Min: 0 Max: 30 Err: 3808
(25.39%) Active: 1000 Started: 1000 Finished: 0
summary = 86590 in 00:03:29 = 413.8/s Avg: 0 Min: 0 Max: 63 Err:
19350 (22.35%)
summary + 15051 in 00:00:30 = 501.5/s Avg: 0 Min: 0 Max: 25 Err: 3827
(25.43%) Active: 1000 Started: 1000 Finished: 0
summary = 101641 in 00:03:59 = 424.8/s Avg: 0 Min: 0 Max: 63 Err:
23177 (22.80%)
summary + 14999 in 00:00:30 = 500.2/s Avg: 0 Min: 0 Max: 29 Err: 3804
(25.36%) Active: 1000 Started: 1000 Finished: 0
summary = 116640 in 00:04:29 = 433.2/s Avg: 0 Min: 0 Max: 63 Err:
26981 (23.13%)
summary + 15007 in 00:00:30 = 500.1/s Avg: 0 Min: 0 Max: 26 Err: 3832
(25.53%) Active: 1000 Started: 1000 Finished: 0
summary = 131647 in 00:04:59 = 439.9/s Avg: 0 Min: 0 Max: 63 Err:
30813 (23.41%)
summary + 15004 in 00:00:30 = 500.2/s Avg: 0 Min: 0 Max: 27 Err: 3836
(25.57%) Active: 1000 Started: 1000 Finished: 0
summary = 146651 in 00:05:29 = 445.4/s Avg: 0 Min: 0 Max: 63 Err:
34649 (23.63%)
summary + 14987 in 00:00:30 = 499.6/s Avg: 0 Min: 0 Max: 29 Err: 3806
(25.40%) Active: 1000 Started: 1000 Finished: 0
summary = 161638 in 00:05:59 = 450.0/s Avg: 0 Min: 0 Max: 63 Err:
38455 (23.79%)
summary + 15034 in 00:00:30 = 501.1/s Avg: 0 Min: 0 Max: 26 Err: 3852
(25.62%) Active: 1000 Started: 1000 Finished: 0
summary = 176672 in 00:06:29 = 453.9/s Avg: 0 Min: 0 Max: 63 Err:
42307 (23.95%)
summary + 14964 in 00:00:30 = 498.8/s Avg: 0 Min: 0 Max: 27 Err: 3801
(25.40%) Active: 1000 Started: 1000 Finished: 0
summary = 191636 in 00:06:59 = 457.1/s Avg: 0 Min: 0 Max: 63 Err:
46108 (24.06%)
summary + 14970 in 00:00:30 = 499.0/s Avg: 0 Min: 0 Max: 28 Err: 3813
(25.47%) Active: 1000 Started: 1000 Finished: 0
summary = 206606 in 00:07:29 = 459.9/s Avg: 0 Min: 0 Max: 63 Err:
49921 (24.16%)
summary + 14980 in 00:00:30 = 499.3/s Avg: 0 Min: 0 Max: 26 Err: 3810
(25.43%) Active: 1000 Started: 1000 Finished: 0

summary = 221586 in 00:07:59 = 462.4/s Avg: 0 Min: 0 Max: 63 Err: 53731 (24.25%)
summary + 14946 in 00:00:30 = 498.1/s Avg: 0 Min: 0 Max: 27 Err: 3811 (25.50%) Active: 1000 Started: 1000 Finished: 0
summary = 236532 in 00:08:29 = 464.5/s Avg: 0 Min: 0 Max: 63 Err: 57542 (24.33%)
summary + 15028 in 00:00:30 = 501.1/s Avg: 0 Min: 0 Max: 29 Err: 3833 (25.51%) Active: 1000 Started: 1000 Finished: 0
summary = 251560 in 00:08:59 = 466.5/s Avg: 0 Min: 0 Max: 63 Err: 61375 (24.40%)
summary + 15002 in 00:00:30 = 500.0/s Avg: 0 Min: 0 Max: 27 Err: 3813 (25.42%) Active: 1000 Started: 1000 Finished: 0
summary = 266562 in 00:09:29 = 468.3/s Avg: 0 Min: 0 Max: 63 Err: 65188 (24.46%)
summary + 15000 in 00:00:30 = 500.0/s Avg: 0 Min: 0 Max: 26 Err: 3799 (25.33%) Active: 1000 Started: 1000 Finished: 0
summary = 281562 in 00:09:59 = 469.9/s Avg: 0 Min: 0 Max: 63 Err: 68987 (24.50%)
summary + 3113 in 00:00:06 = 482.7/s Avg: 0 Min: 0 Max: 25 Err: 514 (16.51%) Active: 0 Started: 1000 Finished: 1000
summary = 284675 in 00:10:06 = 470.0/s Avg: 0 Min: 0 Max: 63 Err: 69501 (24.41%)
Tidying up ... @ 2025 Mar 28 10:14:36 UTC (1743156876451)
... end of run



g.



h. Analyze: The JMeter HTML report shows that out of 284,687 requests, approximately 24.38% failed, primarily due to the server at 172.31.1.46:80 not responding (`NoHttpResponseException`) and a few connection resets. Despite a high throughput of ~475 transactions per second and extremely fast average response times for successful requests (0.70 ms), the high error rate suggests the Nginx server is hitting system or configuration limits under stress, such as too many open connections, insufficient worker threads, or exhausted TCP resources. To improve stability under load, tuning Nginx and OS-level TCP/socket settings, increasing file descriptor limits, and potentially reducing thread count or enabling static content caching are recommended.

i. Next, test with 500 threads instead of 1000 to confirm if Nginx is the bottleneck. Because in the previous test, the total requests were very high: 1000 users × 500 loops = 500,000 requests

2. Static html + 500 users

a. Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Static HTML Stress Test Plan" enabled="true">
      <stringProp name="TestPlan.comments">Stress test static HTML with 500 concurrent users</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
```

```
    <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
      <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
          <stringProp name="Argument.name">HOST</stringProp>
          <stringProp name="Argument.value">172.31.1.46</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
          <stringProp name="Argument.name">PORT</stringProp>
          <stringProp name="Argument.value">80</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
          <stringProp name="Argument.name">PROTOCOL</stringProp>
          <stringProp name="Argument.value">http</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
      </collectionProp>
    </elementProp>
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
  <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Static HTML Stress Group" enabled="true">
      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">500</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /index.html" enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
```

```
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/index.html</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp name="HTTPSampler.response_timeout">30000</stringProp>
       </HTTPSamplerProxy>
       <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK" enabled="true">
         <collectionProp name="Asserion.test_strings">
          <stringProp name="49586">200</stringProp>
         </collectionProp>
         <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
         <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time &lt; 1000ms"
enabled="true">
         <stringProp name="DurationAssertion.duration">1000</stringProp>
        </DurationAssertion>
        <hashTree/>
       </hashTree>
       <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="User Timing Delay"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
       </UniformRandomTimer>
       <hashTree/>
     </hashTree>
    </hashTree>
  </hashTree>
</jmeterTestPlan>
```

b. Run the test
jmeter -n -t test-static-stress-500.jmx -l test-static-stress-500.jtl -e -o test-static-stress-500-result

c. Result
root@ip-172-31-15-56:~/test-0326/user-stress-tests# jmeter -n -t test-static-stress-500.jmx -l test-static-stress-500.jtl -e -o test-static-stress-500-result
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-static-stress-500.jmx
Starting standalone test @ 2025 Mar 28 11:09:53 UTC (1743160193742)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 1 in 00:00:06 = 0.2/s Avg: 59 Min: 59 Max: 59 Err: 0 (0.00%) Active: 12 Started: 12 Finished: 0
summary + 1897 in 00:00:29 = 64.4/s Avg: 1 Min: 0 Max: 16 Err: 0 (0.00%) Active: 257 Started: 257 Finished: 0
summary = 1898 in 00:00:36 = 52.9/s Avg: 1 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 5611 in 00:00:30 = 187.0/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 7509 in 00:01:06 = 114.0/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7516 in 00:00:30 = 250.5/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 15025 in 00:01:36 = 156.7/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7490 in 00:00:30 = 249.7/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 22515 in 00:02:06 = 178.9/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7498 in 00:00:30 = 249.9/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 30013 in 00:02:36 = 192.6/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)

summary + 7536 in 00:00:30 = 251.2/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 37549 in 00:03:06 = 202.0/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7501 in 00:00:30 = 250.0/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 45050 in 00:03:36 = 208.7/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7496 in 00:00:30 = 249.9/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 52546 in 00:04:06 = 213.7/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7513 in 00:00:30 = 250.4/s Avg: 0 Min: 0 Max: 4 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 60059 in 00:04:36 = 217.7/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7469 in 00:00:30 = 249.0/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 67528 in 00:05:06 = 220.8/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7519 in 00:00:30 = 250.6/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 75047 in 00:05:36 = 223.4/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7512 in 00:00:30 = 250.4/s Avg: 0 Min: 0 Max: 14 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 82559 in 00:06:06 = 225.7/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7497 in 00:00:30 = 249.9/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 90056 in 00:06:36 = 227.5/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7506 in 00:00:30 = 250.2/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 97562 in 00:07:06 = 229.1/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7445 in 00:00:30 = 248.1/s Avg: 0 Min: 0 Max: 14 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 105007 in 00:07:36 = 230.3/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7509 in 00:00:30 = 250.3/s Avg: 0 Min: 0 Max: 5 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 112516 in 00:08:06 = 231.6/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)

summary + 7491 in 00:00:30 = 249.7/s Avg: 0 Min: 0 Max: 14 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 120007 in 00:08:36 = 232.6/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7500 in 00:00:30 = 250.0/s Avg: 0 Min: 0 Max: 13 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 127507 in 00:09:06 = 233.6/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7489 in 00:00:30 = 249.7/s Avg: 0 Min: 0 Max: 14 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 134996 in 00:09:36 = 234.4/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
summary + 7369 in 00:00:30 = 249.7/s Avg: 0 Min: 0 Max: 14 Err: 0 (0.00%) Active: 0 Started: 500 Finished: 500
summary = 142365 in 00:10:05 = 235.2/s Avg: 0 Min: 0 Max: 59 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 28 11:19:59 UTC (1743160799510)
Error generating the report:
org.apache.jmeter.report.dashboard.GenerationException: Error while processing samples: Consumer failed with message :Consumer failed with message :Consumer failed with message :Consumer failed with message :Consumer failed with message :Mismatch between expected number of columns:17 and columns in CSV file:6, check your jmeter.save.saveservice.* configuration or check if line 25310 in 'temp/latencyVsRequest/latencyVsRequest2096994654720906687-0' is complete

d. Reason for error: disk full (result file took too much disk space)

oot@ip-172-31-15-56:~# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/root 6.8G 6.7G 6.6M 100% /
tmpfs 479M 0 479M 0% /dev/shm
tmpfs 192M 872K 191M 1% /run
tmpfs 5.0M 0 5.0M 0% /run/lock
/dev/xvda16 881M 137M 683M 17% /boot
/dev/xvda15 105M 6.1M 99M 6% /boot/efi
tmpfs 96M 16K 96M 1% /run/user/0

e. How we fix it

i. Since .jtl files occupied a lot of disk space, we migrated the test results to our local environment using scp. Then we cleaned up the test results to free up the disk space. Then we were able to continue testing.

f. Try again for report

root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-static-stress-500.jmx -l test-static-stress-500.jtl -e -o test-static-stress-500-result
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-static-stress-500.jmx
Starting standalone test @ 2025 Mar 28 18:19:09 UTC (1743185949831)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 401 in 00:00:20 = 20.2/s Avg: 1 Min: 0 Max: 63 Err: 0 (0.00%) Active: 124 Started: 124 Finished: 0
summary + 3609 in 00:00:30 = 120.3/s Avg: 1 Min: 0 Max: 16 Err: 0 (0.00%) Active: 374 Started: 374 Finished: 0
summary = 4010 in 00:00:50 = 80.5/s Avg: 1 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 6960 in 00:00:30 = 232.0/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 10970 in 00:01:20 = 137.4/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 7466 in 00:00:30 = 248.8/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 18436 in 00:01:50 = 167.9/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 7505 in 00:00:30 = 250.2/s Avg: 0 Min: 0 Max: 24 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 25941 in 00:02:20 = 185.5/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 7532 in 00:00:30 = 251.1/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 33473 in 00:02:50 = 197.1/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 7516 in 00:00:30 = 250.6/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

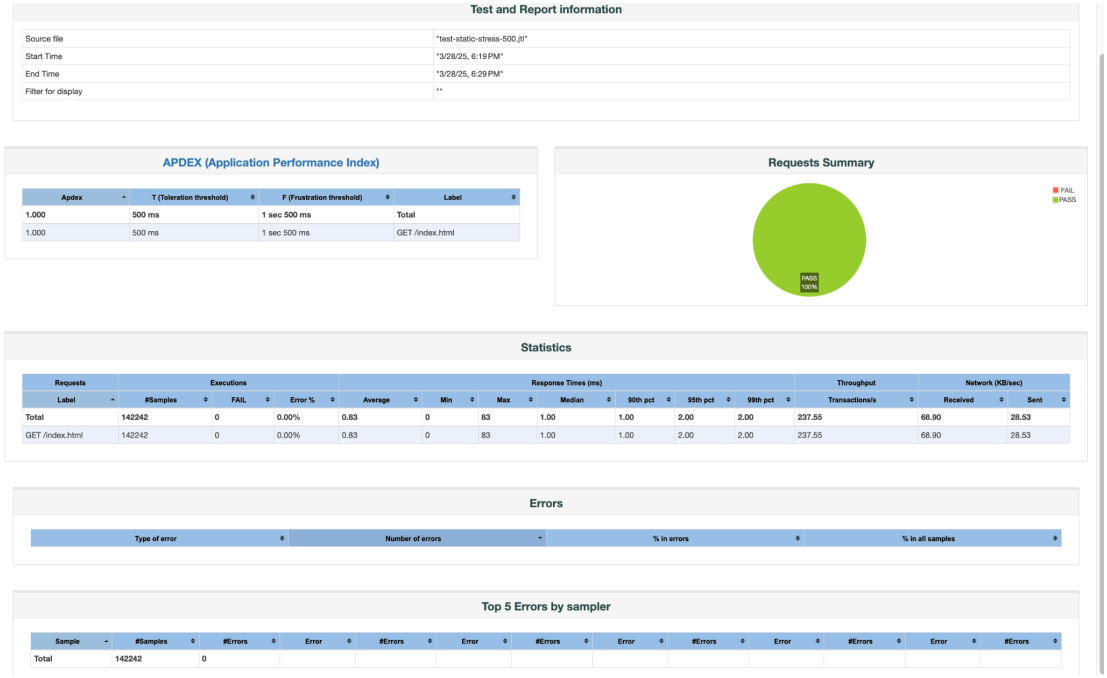
summary = 40989 in 00:03:20 = 205.1/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7506 in 00:00:30 = 250.2/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 48495 in 00:03:50 = 211.0/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7502 in 00:00:30 = 250.0/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 55997 in 00:04:20 = 215.5/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7461 in 00:00:30 = 248.7/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 63458 in 00:04:50 = 218.9/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7502 in 00:00:30 = 250.1/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 70960 in 00:05:20 = 221.9/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7482 in 00:00:30 = 249.4/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 78442 in 00:05:50 = 224.2/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7496 in 00:00:30 = 249.8/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 85938 in 00:06:20 = 226.2/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7408 in 00:00:30 = 247.0/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 93346 in 00:06:50 = 227.8/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7487 in 00:00:30 = 249.6/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 100833 in 00:07:20 = 229.3/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7496 in 00:00:30 = 249.8/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 108329 in 00:07:50 = 230.6/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7521 in 00:00:30 = 250.7/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 115850 in 00:08:20 = 231.8/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7546 in 00:00:30 = 251.6/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 123396 in 00:08:50 = 232.9/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7481 in 00:00:30 = 249.4/s Avg: 0 Min: 0 Max: 14 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 130877 in 00:09:20 = 233.8/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 7529 in 00:00:30 = 251.0/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0
summary = 138406 in 00:09:50 = 234.7/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
summary + 3836 in 00:00:16 = 246.0/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 0 Started: 500 Finished: 500
summary = 142242 in 00:10:05 = 234.9/s Avg: 0 Min: 0 Max: 83 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 28 18:29:15 UTC (1743186555600)
... end of run

g. View the report

root@ip-172-31-15-56:~/test-0328/test-static-stress-500-result# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
144.91.51.56 - - [28/Mar/2025 18:42:07] "GET / HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /sbadmin2-1.0.7/bower_components/bootstrap/dist/css/bootstrap.min.css HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /sbadmin2-1.0.7/bower_components/metisMenu/dist/metisMenu.min.css HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /sbadmin2-1.0.7/dist/css/sb-admin-2.css HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /sbadmin2-1.0.7/bower_components/font-awesome/css/font-awesome.min.css HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /content/css/dashboard.css HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /content/css/theme.blue.css HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /sbadmin2-1.0.7/bower_components/jquery/dist/jquery.min.js HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /sbadmin2-1.0.7/bower_components/bootstrap/dist/js/bootstrap.min.js HTTP/1.1" 200 -

144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/flot/excanvas.min.js HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/flot/jquery.flot.pie.js HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/flot/jquery.flot.resize.js HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/flot/jquery.flot.js HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/flot/jquery.flot.time.js HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/flot.tooltip/js/jquery.flot.tooltip.min.js
HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/flot-axislabels/jquery.flot.axislabels.js
HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/metisMenu/dist/metisMenu.min.js
HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/bower_components/font-awesome/fonts/fontawesome-webfont.
woff?v=4.2.0 HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /content/js/dashboard-commons.js
HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /content/js/dashboard.js
HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET
/sbadmin2-1.0.7/dist/js/sb-admin-2.js HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /content/js/jquery.tablesorter.min.js
HTTP/1.1" 200 -
144.91.51.56 - - [28/Mar/2025 18:42:08] "GET /content/pages/icon-apache.png
HTTP/1.1" 200 -



Test and Report information

Source file	"test-static-stress-500.jtl"
Start Time	"3/28/25, 6:19 PM"
End Time	"3/28/25, 6:29 PM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	500 ms	1 sec 500 ms	Total
1.000	500 ms	1 sec 500 ms	GET /index.html

Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	142242	0	0.00%	0.83	0	83	1.00	1.00	2.00	2.00	237.55	68.90	28.53
GET /index.html	142242	0	0.00%	0.83	0	83	1.00	1.00	2.00	2.00	237.55	68.90	28.53

Errors

Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	142242	0										

h. Analyze

The stress test with 500 concurrent users on the static HTML page completed successfully with 0% error rate, excellent response times (average: 0.83 ms, 99th percentile: 2 ms), and perfect APDEX score of 1.000, confirming that the server handles this load very efficiently. Based on these findings—and considering that the 1000-user test previously showed a 24% error rate—we will now proceed with a **750-user** stress test to pinpoint the upper stability threshold and observe when performance begins to degrade.

3. Static html + 750 users

a. Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Static HTML Stress Test Plan" enabled="true">
      <stringProp name="TestPlan.comments">Stress test static HTML with 500 concurrent users</stringProp>
```

```
    <boolProp name="TestPlan.functional_mode">false</boolProp>
    <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
    <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
      <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
          <stringProp name="Argument.name">HOST</stringProp>
          <stringProp name="Argument.value">172.31.1.46</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
          <stringProp name="Argument.name">PORT</stringProp>
          <stringProp name="Argument.value">80</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
          <stringProp name="Argument.name">PROTOCOL</stringProp>
          <stringProp name="Argument.value">http</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
      </collectionProp>
    </elementProp>
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
  <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Static HTML Stress Group" enabled="true">
      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">750</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /index.html" enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
```

```
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/index.html</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp name="HTTPSampler.response_timeout">30000</stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time &amp;lt; 1000ms"
enabled="true">
          <stringProp name="DurationAssertion.duration">1000</stringProp>
        </DurationAssertion>
        <hashTree/>
      </hashTree>
      <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="User Timing Delay"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
      </UniformRandomTimer>
      <hashTree/>
    </hashTree>
  </hashTree>
</hashTree>
```

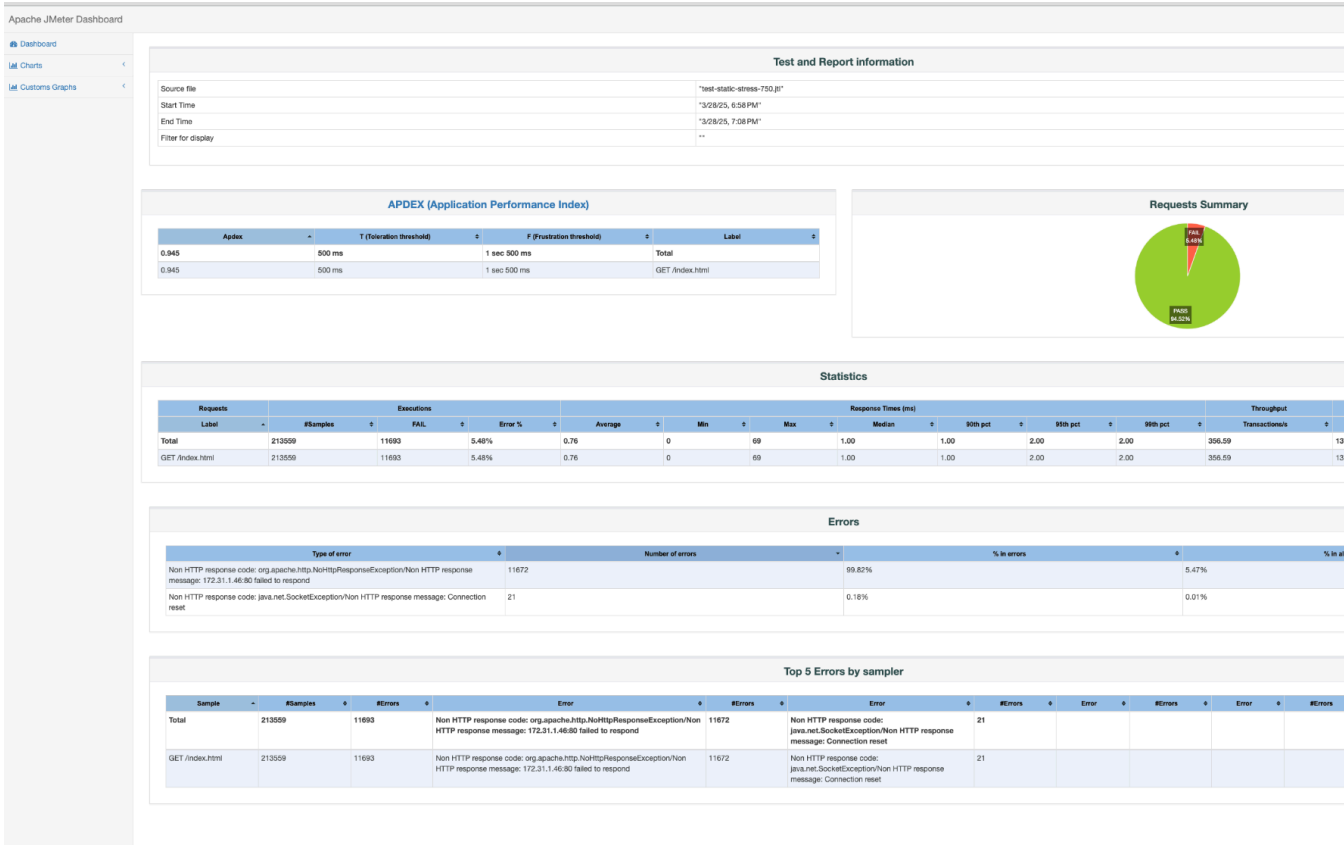
</jmeterTestPlan>

b. Run the test
jmeter -n -t test-static-stress-750.jmx -l test-static-stress-750.jtl -e -o test-static-stress-750-result

c. Result
root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-static-stress-750.jmx -l test-static-stress-750.jtl -e -o test-static-stress-750-result
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-static-stress-750.jmx
Starting standalone test @ 2025 Mar 28 18:58:01 UTC (1743188281126)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 1571 in 00:00:28 = 55.2/s Avg: 1 Min: 0 Max: 69 Err: 0 (0.00%) Active: 294 Started: 294 Finished: 0
summary + 7056 in 00:00:30 = 235.2/s Avg: 0 Min: 0 Max: 20 Err: 0 (0.00%) Active: 668 Started: 668 Finished: 0
summary = 8627 in 00:00:58 = 147.5/s Avg: 0 Min: 0 Max: 69 Err: 0 (0.00%)
summary + 11106 in 00:00:30 = 370.2/s Avg: 0 Min: 0 Max: 28 Err: 536 (4.83%) Active: 750 Started: 750 Finished: 0
summary = 19733 in 00:01:28 = 223.0/s Avg: 0 Min: 0 Max: 69 Err: 536 (2.72%)
summary + 11226 in 00:00:30 = 374.2/s Avg: 0 Min: 0 Max: 29 Err: 638 (5.68%) Active: 750 Started: 750 Finished: 0
summary = 30959 in 00:01:58 = 261.3/s Avg: 0 Min: 0 Max: 69 Err: 1174 (3.79%)
summary + 11281 in 00:00:30 = 376.0/s Avg: 0 Min: 0 Max: 20 Err: 658 (5.83%) Active: 750 Started: 750 Finished: 0
summary = 42240 in 00:02:28 = 284.5/s Avg: 0 Min: 0 Max: 69 Err: 1832 (4.34%)
summary + 11248 in 00:00:30 = 375.0/s Avg: 0 Min: 0 Max: 24 Err: 650 (5.78%) Active: 750 Started: 750 Finished: 0
summary = 53488 in 00:02:58 = 299.7/s Avg: 0 Min: 0 Max: 69 Err: 2482 (4.64%)

summary + 11247 in 00:00:30 = 374.9/s Avg: 0 Min: 0 Max: 21 Err: 641 (5.70%) Active: 750 Started: 750 Finished: 0
summary = 64735 in 00:03:28 = 310.5/s Avg: 0 Min: 0 Max: 69 Err: 3123 (4.82%)
summary + 11202 in 00:00:30 = 373.4/s Avg: 0 Min: 0 Max: 20 Err: 658 (5.87%) Active: 750 Started: 750 Finished: 0
summary = 75937 in 00:03:58 = 318.4/s Avg: 0 Min: 0 Max: 69 Err: 3781 (4.98%)
summary + 11238 in 00:00:30 = 374.6/s Avg: 0 Min: 0 Max: 27 Err: 665 (5.92%) Active: 750 Started: 750 Finished: 0
summary = 87175 in 00:04:28 = 324.7/s Avg: 0 Min: 0 Max: 69 Err: 4446 (5.10%)
summary + 11253 in 00:00:30 = 375.1/s Avg: 0 Min: 0 Max: 21 Err: 632 (5.62%) Active: 750 Started: 750 Finished: 0
summary = 98428 in 00:04:58 = 329.8/s Avg: 0 Min: 0 Max: 69 Err: 5078 (5.16%)
summary + 11285 in 00:00:30 = 376.0/s Avg: 0 Min: 0 Max: 19 Err: 664 (5.88%) Active: 750 Started: 750 Finished: 0
summary = 109713 in 00:05:28 = 334.0/s Avg: 0 Min: 0 Max: 69 Err: 5742 (5.23%)
summary + 11268 in 00:00:30 = 375.7/s Avg: 0 Min: 0 Max: 20 Err: 664 (5.89%) Active: 750 Started: 750 Finished: 0
summary = 120981 in 00:05:58 = 337.5/s Avg: 0 Min: 0 Max: 69 Err: 6406 (5.30%)
summary + 11211 in 00:00:30 = 373.7/s Avg: 0 Min: 0 Max: 20 Err: 645 (5.75%) Active: 750 Started: 750 Finished: 0
summary = 132192 in 00:06:28 = 340.3/s Avg: 0 Min: 0 Max: 69 Err: 7051 (5.33%)
summary + 11291 in 00:00:30 = 376.3/s Avg: 0 Min: 0 Max: 22 Err: 645 (5.71%) Active: 750 Started: 750 Finished: 0
summary = 143483 in 00:06:58 = 342.9/s Avg: 0 Min: 0 Max: 69 Err: 7696 (5.36%)
summary + 11232 in 00:00:30 = 374.4/s Avg: 0 Min: 0 Max: 21 Err: 660 (5.88%) Active: 750 Started: 750 Finished: 0
summary = 154715 in 00:07:28 = 345.0/s Avg: 0 Min: 0 Max: 69 Err: 8356 (5.40%)
summary + 11217 in 00:00:30 = 373.9/s Avg: 0 Min: 0 Max: 21 Err: 637 (5.68%) Active: 750 Started: 750 Finished: 0
summary = 165932 in 00:07:58 = 346.8/s Avg: 0 Min: 0 Max: 69 Err: 8993 (5.42%)
summary + 11239 in 00:00:30 = 374.6/s Avg: 0 Min: 0 Max: 18 Err: 653 (5.81%) Active: 750 Started: 750 Finished: 0
summary = 177171 in 00:08:28 = 348.4/s Avg: 0 Min: 0 Max: 69 Err: 9646 (5.44%)

summary + 11253 in 00:00:30 = 375.1/s Avg: 0 Min: 0 Max: 20 Err: 662 (5.88%) Active: 750 Started: 750 Finished: 0
summary = 188424 in 00:08:58 = 349.9/s Avg: 0 Min: 0 Max: 69 Err: 10308 (5.47%)
summary + 11266 in 00:00:30 = 375.5/s Avg: 0 Min: 0 Max: 22 Err: 659 (5.85%) Active: 750 Started: 750 Finished: 0
summary = 199690 in 00:09:28 = 351.3/s Avg: 0 Min: 0 Max: 69 Err: 10967 (5.49%)
summary + 11285 in 00:00:30 = 376.2/s Avg: 0 Min: 0 Max: 19 Err: 632 (5.60%) Active: 750 Started: 750 Finished: 0
summary = 210975 in 00:09:58 = 352.5/s Avg: 0 Min: 0 Max: 69 Err: 11599 (5.50%)
summary + 2584 in 00:00:07 = 364.3/s Avg: 0 Min: 0 Max: 3 Err: 94 (3.64%) Active: 0 Started: 750 Finished: 750
summary = 213559 in 00:10:06 = 352.7/s Avg: 0 Min: 0 Max: 69 Err: 11693 (5.48%)
Tidying up ... @ 2025 Mar 28 19:08:07 UTC (1743188887095)
... end of run



d. Analyze
The 750-user static HTML stress test completed with a total of 213,559 requests, achieving a high throughput of 356.59 transactions per second, but with a notable

error rate of 5.47%—primarily caused by connection issues (NoHttpResponseException and SocketException). Despite a strong APDEX score of 0.945 and excellent response times (99th percentile: 2ms, max: 69ms), the system started showing signs of strain under this load. These results confirm that the server is beginning to hit resource or concurrency limits around 750 users. Based on this, we will proceed with a 650-user stress test to better identify the highest stable concurrency threshold before errors begin to significantly increase.

4. Static html + 650 Users

a. Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Static
HTML Stress Test Plan" enabled="true">
      <stringProp name="TestPlan.comments">Stress test static HTML with 500
concurrent users</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
        <collectionProp name="Arguments.arguments">
          <elementProp name="HOST" elementType="Argument">
            <stringProp name="Argument.name">HOST</stringProp>
            <stringProp name="Argument.value">172.31.1.46</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PORT" elementType="Argument">
            <stringProp name="Argument.name">PORT</stringProp>
            <stringProp name="Argument.value">80</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PROTOCOL" elementType="Argument">
            <stringProp name="Argument.name">PROTOCOL</stringProp>
            <stringProp name="Argument.value">http</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="TestPlan.user_define_classpath"></stringProp>
```

```
  </TestPlan>
  <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Static HTML Stress Group" enabled="true">
      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">650</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /index.html" enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/index.html</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp name="HTTPSampler.response_timeout">30000</stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
```

```
          <stringProp
        name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
        testclass="DurationAssertion" testname="Response Time &lt; 1000ms"
        enabled="true">
          <stringProp name="DurationAssertion.duration">1000</stringProp>
        </DurationAssertion>
        <hashTree/>
      </hashTree>
      <UniformRandomTimer guiclass="UniformRandomTimerGui"
        testclass="UniformRandomTimer" testname="User Timing Delay"
        enabled="true">
          <stringProp name="ConstantTimer.delay">1000</stringProp>
          <stringProp name="RandomTimer.range">2000</stringProp>
      </UniformRandomTimer>
      <hashTree/>
    </hashTree>
  </hashTree>
 </hashTree>
</jmeterTestPlan>
```

b. Run the test
   jmeter -n -t test-static-stress-650.jmx -l test-static-stress-650.jtl -e -o test-static-stress-650-result

c. Result
   root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-static-stress-650.jmx -l test-static-stress-650.jtl -e -o test-static-stress-650-result
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
   Creating summariser <summary>
   Created the tree successfully using test-static-stress-650.jmx
   Starting standalone test @ 2025 Mar 28 19:39:23 UTC (1743190763987)
   Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445

summary + 1 in 00:00:07 = 0.1/s Avg: 54 Min: 54 Max: 54 Err: 0 (0.00%) Active: 20 Started: 20 Finished: 0
summary + 2370 in 00:00:29 = 82.4/s Avg: 1 Min: 0 Max: 75 Err: 0 (0.00%) Active: 333 Started: 333 Finished: 0
summary = 2371 in 00:00:36 = 66.6/s Avg: 1 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 7324 in 00:00:30 = 244.2/s Avg: 0 Min: 0 Max: 25 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 9695 in 00:01:06 = 147.8/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9788 in 00:00:30 = 326.2/s Avg: 0 Min: 0 Max: 25 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 19483 in 00:01:36 = 203.8/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9757 in 00:00:30 = 325.3/s Avg: 0 Min: 0 Max: 25 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 29240 in 00:02:06 = 232.8/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9748 in 00:00:30 = 324.9/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 38988 in 00:02:36 = 250.6/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9724 in 00:00:30 = 324.1/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 48712 in 00:03:06 = 262.4/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9759 in 00:00:30 = 325.4/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 58471 in 00:03:36 = 271.2/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9741 in 00:00:30 = 324.7/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 68212 in 00:04:06 = 277.7/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9789 in 00:00:30 = 326.3/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 78001 in 00:04:36 = 283.0/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9733 in 00:00:30 = 324.4/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 87734 in 00:05:06 = 287.1/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9800 in 00:00:30 = 326.6/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0

summary = 97534 in 00:05:36 = 290.6/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9783 in 00:00:30 = 326.2/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 107317 in 00:06:06 = 293.5/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9753 in 00:00:30 = 324.9/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 117070 in 00:06:36 = 295.9/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9739 in 00:00:30 = 324.8/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 126809 in 00:07:06 = 298.0/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9762 in 00:00:30 = 325.4/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 136571 in 00:07:36 = 299.8/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9795 in 00:00:30 = 326.5/s Avg: 0 Min: 0 Max: 15 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 146366 in 00:08:06 = 301.4/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9794 in 00:00:30 = 326.4/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 156160 in 00:08:36 = 302.9/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9784 in 00:00:30 = 326.2/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 165944 in 00:09:06 = 304.1/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9752 in 00:00:30 = 325.1/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 650 Started: 650 Finished: 0
summary = 175696 in 00:09:36 = 305.2/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
summary + 9651 in 00:00:30 = 322.9/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 0 Started: 650 Finished: 650
summary = 185347 in 00:10:05 = 306.1/s Avg: 0 Min: 0 Max: 75 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 28 19:49:29 UTC (1743191369895)
... end of run

d. Analyze
The 650-user static HTML stress test completed with excellent results—**100% success rate** and **zero errors** across 185,347 requests over 10 minutes. The

server maintained stable performance with a peak throughput of over 325 requests per second and a max response time of just 75ms, indicating it handled this level of concurrency smoothly without degradation. Given this strong performance, we proceeded to quickly test with 700 users to pinpoint the system's upper stability threshold.

5. Static html + 700 users

a. Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Static
HTML Stress Test Plan" enabled="true">
      <stringProp name="TestPlan.comments">Stress test static HTML with 500
concurrent users</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
        <collectionProp name="Arguments.arguments">
          <elementProp name="HOST" elementType="Argument">
            <stringProp name="Argument.name">HOST</stringProp>
            <stringProp name="Argument.value">172.31.1.46</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PORT" elementType="Argument">
            <stringProp name="Argument.name">PORT</stringProp>
            <stringProp name="Argument.value">80</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PROTOCOL" elementType="Argument">
            <stringProp name="Argument.name">PROTOCOL</stringProp>
            <stringProp name="Argument.value">http</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="TestPlan.user_define_classpath"></stringProp>
    </TestPlan>
    <hashTree>
      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Static HTML Stress Group" enabled="true">
```

```
      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">700</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /index.html" enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/index.html</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp name="HTTPSampler.response_timeout">30000</stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK" enabled="true">
          <collectionProp name="Asserion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
```

```
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time &lt; 1000ms"
enabled="true">
          <stringProp name="DurationAssertion.duration">1000</stringProp>
        </DurationAssertion>
        <hashTree/>
       </hashTree>
       <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="User Timing Delay"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
       </UniformRandomTimer>
       <hashTree/>
      </hashTree>
    </hashTree>
  </hashTree>
</jmeterTestPlan>
```

b. Run the test
   jmeter -n -t test-static-stress-700.jmx -l test-static-stress-700.jtl -e -o test-static-stress-700-result
c. Result

```
root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-static-stress-700.jmx -l
test-static-stress-700.jtl -e -o test-static-stress-700-result
WARN StatusConsoleListener The use of package scanning to locate plugins is
deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is
deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is
deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is
deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-static-stress-700.jmx
Starting standalone test @ 2025 Mar 28 19:52:59 UTC (1743191579503)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message
on port 4445
summary +   1711 in 00:00:30 =   56.7/s Avg:     1 Min:     0 Max:    63 Err:     0
(0.00%) Active: 293 Started: 293 Finished: 0
summary +   6854 in 00:00:30 =  228.5/s Avg:     0 Min:     0 Max:    18 Err:     0
(0.00%) Active: 642 Started: 642 Finished: 0
```

summary = 8565 in 00:01:00 = 142.3/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10446 in 00:00:30 = 348.3/s Avg: 0 Min: 0 Max: 24 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 19011 in 00:01:30 = 210.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10471 in 00:00:30 = 348.9/s Avg: 0 Min: 0 Max: 28 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 29482 in 00:02:00 = 245.3/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10495 in 00:00:30 = 350.0/s Avg: 0 Min: 0 Max: 23 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 39977 in 00:02:30 = 266.2/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10474 in 00:00:30 = 349.1/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 50451 in 00:03:00 = 280.0/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10444 in 00:00:30 = 348.1/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 60895 in 00:03:30 = 289.7/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10486 in 00:00:30 = 349.5/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 71381 in 00:04:00 = 297.2/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10459 in 00:00:30 = 348.5/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 81840 in 00:04:30 = 302.9/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10470 in 00:00:30 = 349.0/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 92310 in 00:05:00 = 307.5/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10508 in 00:00:30 = 350.4/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 102818 in 00:05:30 = 311.4/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10482 in 00:00:30 = 349.4/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 113300 in 00:06:00 = 314.6/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10505 in 00:00:30 = 350.2/s Avg: 0 Min: 0 Max: 4 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 123805 in 00:06:30 = 317.3/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10485 in 00:00:30 = 349.5/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 134290 in 00:07:00 = 319.6/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10496 in 00:00:30 = 349.9/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 144786 in 00:07:30 = 321.6/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10563 in 00:00:30 = 352.1/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 155349 in 00:08:00 = 323.5/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10472 in 00:00:30 = 349.1/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 165821 in 00:08:30 = 325.0/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10467 in 00:00:30 = 348.9/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 176288 in 00:09:00 = 326.3/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10536 in 00:00:30 = 351.2/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 186824 in 00:09:30 = 327.7/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 10497 in 00:00:30 = 349.9/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0
summary = 197321 in 00:10:00 = 328.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
summary + 1811 in 00:00:05 = 336.1/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 0 Started: 700 Finished: 700
summary = 199132 in 00:10:06 = 328.8/s Avg: 0 Min: 0 Max: 63 Err: 0 (0.00%)
Tidying up ... @ 2025 Mar 28 20:03:05 UTC (1743192185392)
... end of run

d. Analyze

The 700-user static HTML stress test concluded with excellent result: 199,132 requests completed with 0% errors, all 700 users finished successfully, and the server maintained consistent throughput at ~329 requests per second with low response times (max: 63ms). This confirms that the server can reliably handle up to 700 concurrent users without performance degradation. Since we previously

observed failures starting at 750 users (~5.5% error rate), we can confidently conclude that **the maximum stable concurrency limit for static HTML stress testing is 700 users** under the current configuration.

6. Dynamic PHP + 500 Users

a. Set up PHP on nginx

i. sudo nano /etc/nginx/sites-available/default

ii. Paste the following

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;
    index index.php index.html index.htm;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
        index index.php index.html index.htm;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

iii. Restart Nginx

```
root@ip-172-31-1-46:~# sudo systemctl restart php8.3-fpm
sudo systemctl restart nginx
```

iv. Create a test PHP file

```
root@ip-172-31-1-46:~# echo '<?php echo "PHP is working\!"; ?>' | sudo
tee /var/www/html/test.php
<?php echo "PHP is working\!"; ?>
```

b. Test if works

curl http://localhost/test.php

c. Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan"
testname="Dynamic PHP Stress Test Plan" enabled="true">
      <stringProp name="TestPlan.comments">Stress testing PHP endpoint
with 500 users</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
        <collectionProp name="Arguments.arguments">
          <elementProp name="HOST" elementType="Argument">
            <stringProp name="Argument.name">HOST</stringProp>
            <stringProp name="Argument.value">172.31.1.46</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PORT" elementType="Argument">
            <stringProp name="Argument.name">PORT</stringProp>
            <stringProp name="Argument.value">80</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PROTOCOL" elementType="Argument">
            <stringProp name="Argument.name">PROTOCOL</stringProp>
            <stringProp name="Argument.value">http</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
      <stringProp name="TestPlan.user_define_classpath"></stringProp>
    </TestPlan>
    <hashTree>
```

```
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="PHP Stress Test Group" enabled="true">
      <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp
name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">500</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /test.php"
enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp
name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.path">/test.php</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp
name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp
name="HTTPSampler.response_timeout">30000</stringProp>
```

```
        </HTTPSamplerProxy>
        <hashTree>
          <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Response Code Assertion"
enabled="true">
            <collectionProp name="Asserion.test_strings">
              <stringProp name="49586">200</stringProp>
            </collectionProp>
            <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
            <boolProp name="Assertion.assume_success">false</boolProp>
            <intProp name="Assertion.test_type">8</intProp>
          </ResponseAssertion>
          <hashTree/>
          <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time LT 1000ms"
enabled="true">
            <stringProp
name="DurationAssertion.duration">1000</stringProp>
          </DurationAssertion>
          <hashTree/>
        </hashTree>
        <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Realistic Timing"
enabled="true">
          <stringProp name="ConstantTimer.delay">1000</stringProp>
          <stringProp name="RandomTimer.range">2000</stringProp>
        </UniformRandomTimer>
        <hashTree/>
      </hashTree>
    </hashTree>
  </hashTree>
</jmeterTestPlan>
```

d. Run the test

jmeter -n -t test-dynamic-stress-500.jmx -l result-dynamic-stress-500.jtl -e -o report-dynamic-500

e. Result
   i. root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-dynamic-stress-500.jmx -l result-dynamic-stress-500.jtl -e -o report-dynamic-500

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      Creating summariser <summary>

      Created the tree successfully using test-dynamic-stress-500.jmx

      Starting standalone test @ 2025 Mar 29 01:31:23 UTC (1743211883220)

      Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445

      summary + 1 in 00:00:07 = 0.2/s Avg: 76 Min: 76 Max: 76 Err: 0 (0.00%) Active: 13 Started: 13 Finished: 0

      summary + 1921 in 00:00:30 = 64.4/s Avg: 1 Min: 0 Max: 49 Err: 0 (0.00%) Active: 262 Started: 262 Finished: 0

      summary = 1922 in 00:00:36 = 52.8/s Avg: 1 Min: 0 Max: 76 Err: 0 (0.00%)

      summary + 5658 in 00:00:30 = 188.7/s Avg: 1 Min: 0 Max: 13 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

      summary = 7580 in 00:01:06 = 114.2/s Avg: 1 Min: 0 Max: 76 Err: 0 (0.00%)

      summary + 7491 in 00:00:30 = 249.7/s Avg: 0 Min: 0 Max: 6 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 15071 in 00:01:36 = 156.4/s Avg: 1 Min: 0 Max: 76 Err: 0 (0.00%)

summary + 7567 in 00:00:30 = 252.2/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 22638 in 00:02:06 = 179.1/s Avg: 1 Min: 0 Max: 76 Err: 0 (0.00%)

summary + 7488 in 00:00:30 = 249.6/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 30126 in 00:02:36 = 192.6/s Avg: 0 Min: 0 Max: 76 Err: 0 (0.00%)

summary + 7498 in 00:00:30 = 250.0/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 37624 in 00:03:06 = 201.9/s Avg: 0 Min: 0 Max: 76 Err: 0 (0.00%)

summary + 7469 in 00:00:30 = 249.0/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 45093 in 00:03:36 = 208.4/s Avg: 0 Min: 0 Max: 76 Err: 0 (0.00%)

summary + 7481 in 00:00:30 = 249.4/s Avg: 0 Min: 0 Max: 3 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 52574 in 00:04:06 = 213.4/s Avg: 0 Min: 0 Max: 76 Err: 0 (0.00%)

summary + 7483 in 00:00:30 = 249.4/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 60057 in 00:04:36 = 217.3/s Avg: 0 Min: 0 Max: 76 Err: 0 (0.00%)

summary + 7449 in 00:00:30 = 248.4/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 67506 in 00:05:06 = 220.3/s Avg: 0 Min: 0 Max: 76 Err: 0 (0.00%)

summary + 7515 in 00:00:30 = 250.5/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 75021 in 00:05:36 = 223.0/s Avg: 0 Min: 0 Max:
76 Err: 0 (0.00%)

summary +   7486 in 00:00:30 = 249.5/s Avg: 0 Min: 0 Max:
13 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 82507 in 00:06:06 = 225.2/s Avg: 0 Min: 0 Max:
76 Err: 0 (0.00%)

summary +   7444 in 00:00:30 = 248.2/s Avg: 0 Min: 0 Max:
15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 89951 in 00:06:36 = 226.9/s Avg: 0 Min: 0 Max:
76 Err: 0 (0.00%)

summary +   7474 in 00:00:30 = 249.2/s Avg: 0 Min: 0 Max:
13 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 97425 in 00:07:06 = 228.5/s Avg: 0 Min: 0 Max:
76 Err: 0 (0.00%)

summary +   7469 in 00:00:30 = 249.0/s Avg: 0 Min: 0 Max:
15 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 104894 in 00:07:36 = 229.8/s Avg: 0 Min: 0 Max:
76 Err: 0 (0.00%)

summary +   7503 in 00:00:30 = 250.0/s Avg: 0 Min: 0 Max:
14 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 112397 in 00:08:06 = 231.1/s Avg: 0 Min: 0 Max:
76 Err: 0 (0.00%)

summary +   7491 in 00:00:30 = 249.7/s Avg: 0 Min: 0 Max:
13 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 119888 in 00:08:36 = 232.2/s Avg: 0 Min: 0 Max:
76 Err: 0 (0.00%)

summary +   7476 in 00:00:30 = 249.1/s Avg: 0 Min: 0 Max:
14 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 127364 in 00:09:06 = 233.1/s Avg: 0 Min: 0 Max:
76 Err: 0 (0.00%)

summary +   7494 in 00:00:30 = 249.9/s Avg: 0 Min: 0 Max:
14 Err: 0 (0.00%) Active: 500 Started: 500 Finished: 0

summary = 134858 in 00:09:36 = 234.0/s Avg: 0 Min: 0 Max: 76 Err: 0 (0.00%)

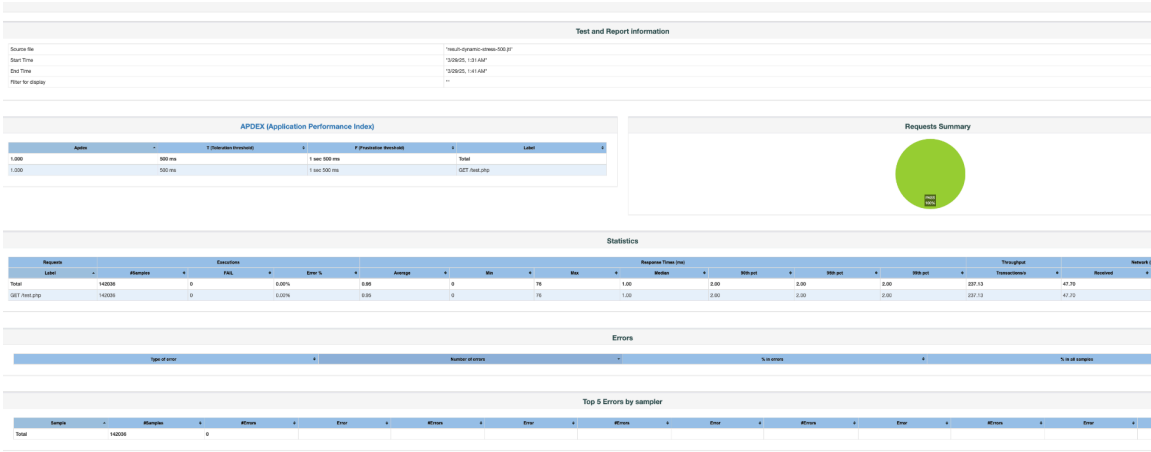
summary + 7178 in 00:00:29 = 246.9/s Avg: 0 Min: 0 Max: 14 Err: 0 (0.00%) Active: 0 Started: 500 Finished: 500

summary = 142036 in 00:10:05 = 234.6/s Avg: 0 Min: 0 Max: 76 Err: 0 (0.00%)

Tidying up ... @ 2025 Mar 29 01:41:29 UTC (1743212489069)

... end of run

ii. View result: python3 -m http.server 8080
Go to: http://54.176.203.205:8080/



7. Dynamic PHP +700 users

a. Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="PHP Stress Test Plan (700 Users)" enabled="true">
      <stringProp name="TestPlan.comments">Stress test /test.php with 700 users</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
```

```
    <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
      <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
          <stringProp name="Argument.name">HOST</stringProp>
          <stringProp name="Argument.value">172.31.1.46</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
          <stringProp name="Argument.name">PORT</stringProp>
          <stringProp name="Argument.value">80</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
          <stringProp name="Argument.name">PROTOCOL</stringProp>
          <stringProp name="Argument.value">http</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
      </collectionProp>
    </elementProp>
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
  <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="PHP Stress Group (700 Users)" enabled="true">
      <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp
name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">700</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
```

```
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /test.php"
enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp
name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.path">/test.php</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp
name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp
name="HTTPSampler.response_timeout">30000</stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK"
enabled="true">
          <collectionProp name="Assertion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time &lt; 2000ms"
enabled="true">
          <stringProp
name="DurationAssertion.duration">2000</stringProp>
```

```
        </DurationAssertion>
        <hashTree/>
      </hashTree>
      <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Random Delay Timer"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
      </UniformRandomTimer>
      <hashTree/>
    </hashTree>
  </hashTree>
 </hashTree>
</jmeterTestPlan>
```

b. Run the test

   jmeter -n -t test-dynamic-stress-700.jmx -l result-dynamic-stress-700.jtl -e -o report-dynamic-700

c. Result
   i. Result

      root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-php-stress-700.jmx -l result-php-stress-700.jtl -e -o report-php-700
      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      An error occurred: The file /root/test-0328/test-php-stress-700.jmx doesn't exist or can't be opened

      root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-dynamic-stress-700.jmx -l result-dynamic-stress-700.jtl -e -o report-dynamic-700

WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

Creating summariser <summary>

Created the tree successfully using test-dynamic-stress-700.jmx

Starting standalone test @ 2025 Mar 29 01:55:58 UTC (1743213358491)

Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445

summary +   1 in 00:00:07 =   0.1/s Avg: 61 Min: 61 Max: 61 Err: 0 (0.00%) Active: 22 Started: 22 Finished: 0

summary +   1842 in 00:00:24 =   75.8/s Avg: 1 Min: 0 Max: 61 Err: 0 (0.00%) Active: 304 Started: 304 Finished: 0

summary =   1843 in 00:00:31 =   59.2/s Avg: 1 Min: 0 Max: 61 Err: 0 (0.00%)

summary +   6991 in 00:00:30 = 233.0/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 653 Started: 653 Finished: 0

summary =   8834 in 00:01:01 = 144.5/s Avg: 1 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10460 in 00:00:30 = 348.7/s Avg: 0 Min: 0 Max: 28 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 19294 in 00:01:31 = 211.7/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10433 in 00:00:30 = 347.8/s Avg: 0 Min: 0 Max: 28 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 29727 in 00:02:01 = 245.4/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10503 in 00:00:30 = 350.1/s Avg: 0 Min: 0 Max: 23 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 40230 in 00:02:31 = 266.2/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10470 in 00:00:30 = 348.9/s Avg: 0 Min: 0 Max: 22 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 50700 in 00:03:01 = 279.9/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10516 in 00:00:30 = 350.6/s Avg: 0 Min: 0 Max: 20 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 61216 in 00:03:31 = 289.9/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10487 in 00:00:30 = 349.5/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 71703 in 00:04:01 = 297.3/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10459 in 00:00:30 = 348.7/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 82162 in 00:04:31 = 303.0/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10526 in 00:00:30 = 350.9/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 92688 in 00:05:01 = 307.8/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10487 in 00:00:30 = 349.6/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 103175 in 00:05:31 = 311.6/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10514 in 00:00:30 = 350.4/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 113689 in 00:06:01 = 314.8/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10492 in 00:00:30 = 349.8/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 124181 in 00:06:31 = 317.5/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10554 in 00:00:30 = 351.8/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 134735 in 00:07:01 = 319.9/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10509 in 00:00:30 = 350.3/s Avg: 0 Min: 0 Max: 18 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 145244 in 00:07:31 = 321.9/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10499 in 00:00:30 = 350.0/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 155743 in 00:08:01 = 323.7/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10491 in 00:00:30 = 349.7/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 166234 in 00:08:31 = 325.2/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10517 in 00:00:30 = 350.6/s Avg: 0 Min: 0 Max: 17 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 176751 in 00:09:01 = 326.6/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10535 in 00:00:30 = 351.2/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 187286 in 00:09:31 = 327.9/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 10513 in 00:00:30 = 350.4/s Avg: 0 Min: 0 Max: 19 Err: 0 (0.00%) Active: 700 Started: 700 Finished: 0

summary = 197799 in 00:10:01 = 329.0/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

summary + 1470 in 00:00:04 = 334.2/s Avg: 0 Min: 0 Max: 16 Err: 0 (0.00%) Active: 0 Started: 700 Finished: 700

summary = 199269 in 00:10:06 = 329.1/s Avg: 0 Min: 0 Max: 61 Err: 0 (0.00%)

Tidying up ... @ 2025 Mar 29 02:06:04 UTC (1743213964403)

... end of run

ii. View result: python3 -m http.server 8080
Go to: http://54.176.203.205:8080/

8. Dynamic PHP + 750 Users

a. Configurations

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="PHP Stress Test Plan (750 Users)" enabled="true">
      <stringProp name="TestPlan.comments">Stress test /test.php with 750 users</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables" elementType="Arguments">
        <collectionProp name="Arguments.arguments">
          <elementProp name="HOST" elementType="Argument">
            <stringProp name="Argument.name">HOST</stringProp>
            <stringProp name="Argument.value">172.31.1.46</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PORT" elementType="Argument">
            <stringProp name="Argument.name">PORT</stringProp>
            <stringProp name="Argument.value">80</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PROTOCOL" elementType="Argument">
```

```
          <stringProp name="Argument.name">PROTOCOL</stringProp>
          <stringProp name="Argument.value">http</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
      </collectionProp>
    </elementProp>
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
  <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="PHP Stress Group (750 Users)" enabled="true">
      <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp
name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">750</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /test.php"
enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp
name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.path">/test.php</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
```

```
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp
name="HTTPSampler.connect_timeout">5000</stringProp>
        <stringProp
name="HTTPSampler.response_timeout">30000</stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK"
enabled="true">
          <collectionProp name="Assertion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
        <DurationAssertion guiclass="DurationAssertionGui"
testclass="DurationAssertion" testname="Response Time &amp;lt; 2000ms"
enabled="true">
          <stringProp
name="DurationAssertion.duration">2000</stringProp>
        </DurationAssertion>
        <hashTree/>
      </hashTree>
      <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Random Delay Timer"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
      </UniformRandomTimer>
      <hashTree/>
    </hashTree>
  </hashTree>
</hashTree>
</jmeterTestPlan>
```

d. Run the test

   jmeter -n -t test-dynamic-stress-750.jmx -l result-dynamic-stress-750.jtl -e -o report-dynamic-750

e. Result
   i. root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-dynamic-stress-750.jmx -l result-dynamic-stress-750.jtl -e -o report-dynamic-750

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

      Creating summariser <summary>

      Created the tree successfully using test-dynamic-stress-750.jmx

      Starting standalone test @ 2025 Mar 29 02:13:27 UTC (1743214407020)

      Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445

      summary + 1 in 00:00:07 = 0.1/s Avg: 11 Min: 11 Max: 11 Err: 0 (0.00%) Active: 22 Started: 22 Finished: 0

      summary + 2205 in 00:00:26 = 85.4/s Avg: 1 Min: 0 Max: 68 Err: 0 (0.00%) Active: 345 Started: 345 Finished: 0

      summary = 2206 in 00:00:33 = 67.7/s Avg: 1 Min: 0 Max: 68 Err: 0 (0.00%)

      summary + 7792 in 00:00:30 = 259.7/s Avg: 0 Min: 0 Max: 21 Err: 0 (0.00%) Active: 720 Started: 720 Finished: 0

summary = 9998 in 00:01:03 = 159.8/s Avg: 1 Min: 0 Max: 68 Err: 0 (0.00%)

summary + 11231 in 00:00:30 = 374.4/s Avg: 0 Min: 0 Max: 29 Err: 639 (5.69%) Active: 750 Started: 750 Finished: 0

summary = 21229 in 00:01:33 = 229.3/s Avg: 0 Min: 0 Max: 68 Err: 639 (3.01%)

summary + 11214 in 00:00:30 = 373.8/s Avg: 0 Min: 0 Max: 22 Err: 675 (6.02%) Active: 750 Started: 750 Finished: 0

summary = 32443 in 00:02:03 = 264.7/s Avg: 0 Min: 0 Max: 68 Err: 1314 (4.05%)

summary + 11208 in 00:00:30 = 373.6/s Avg: 0 Min: 0 Max: 26 Err: 662 (5.91%) Active: 750 Started: 750 Finished: 0

summary = 43651 in 00:02:33 = 286.1/s Avg: 0 Min: 0 Max: 68 Err: 1976 (4.53%)

summary + 11257 in 00:00:30 = 375.2/s Avg: 0 Min: 0 Max: 20 Err: 692 (6.15%) Active: 750 Started: 750 Finished: 0

summary = 54908 in 00:03:03 = 300.7/s Avg: 0 Min: 0 Max: 68 Err: 2668 (4.86%)

summary + 11214 in 00:00:30 = 373.8/s Avg: 0 Min: 0 Max: 20 Err: 668 (5.96%) Active: 750 Started: 750 Finished: 0

summary = 66122 in 00:03:33 = 311.1/s Avg: 0 Min: 0 Max: 68 Err: 3336 (5.05%)

summary + 11258 in 00:00:30 = 375.3/s Avg: 0 Min: 0 Max: 19 Err: 680 (6.04%) Active: 750 Started: 750 Finished: 0

summary = 77380 in 00:04:03 = 319.0/s Avg: 0 Min: 0 Max: 68 Err: 4016 (5.19%)

summary + 11189 in 00:00:30 = 373.0/s Avg: 0 Min: 0 Max: 21 Err: 676 (6.04%) Active: 750 Started: 750 Finished: 0

summary = 88569 in 00:04:33 = 324.9/s Avg: 0 Min: 0 Max: 68 Err: 4692 (5.30%)

summary + 11281 in 00:00:30 = 376.0/s Avg: 0 Min: 0 Max: 18 Err: 662 (5.87%) Active: 750 Started: 750 Finished: 0

summary = 99850 in 00:05:03 = 330.0/s Avg: 0 Min: 0 Max:
68 Err: 5354 (5.36%)

summary + 11286 in 00:00:30 = 376.3/s Avg: 0 Min: 0 Max:
18 Err: 693 (6.14%) Active: 750 Started: 750 Finished: 0

summary = 111136 in 00:05:33 = 334.2/s Avg: 0 Min: 0 Max:
68 Err: 6047 (5.44%)

summary + 11173 in 00:00:30 = 372.4/s Avg: 0 Min: 0 Max:
18 Err: 662 (5.92%) Active: 750 Started: 750 Finished: 0

summary = 122309 in 00:06:03 = 337.3/s Avg: 0 Min: 0 Max:
68 Err: 6709 (5.49%)

summary + 11281 in 00:00:30 = 376.1/s Avg: 0 Min: 0 Max:
18 Err: 699 (6.20%) Active: 750 Started: 750 Finished: 0

summary = 133590 in 00:06:33 = 340.3/s Avg: 0 Min: 0 Max:
68 Err: 7408 (5.55%)

summary + 11245 in 00:00:30 = 374.8/s Avg: 0 Min: 0 Max:
25 Err: 666 (5.92%) Active: 750 Started: 750 Finished: 0

summary = 144835 in 00:07:03 = 342.7/s Avg: 0 Min: 0 Max:
68 Err: 8074 (5.57%)

summary + 11247 in 00:00:30 = 374.9/s Avg: 0 Min: 0 Max:
19 Err: 665 (5.91%) Active: 750 Started: 750 Finished: 0

summary = 156082 in 00:07:33 = 344.9/s Avg: 0 Min: 0 Max:
68 Err: 8739 (5.60%)

summary + 11238 in 00:00:30 = 374.6/s Avg: 0 Min: 0 Max:
19 Err: 674 (6.00%) Active: 750 Started: 750 Finished: 0

summary = 167320 in 00:08:03 = 346.7/s Avg: 0 Min: 0 Max:
68 Err: 9413 (5.63%)

summary + 11216 in 00:00:30 = 373.8/s Avg: 0 Min: 0 Max:
20 Err: 685 (6.11%) Active: 750 Started: 750 Finished: 0

summary = 178536 in 00:08:33 = 348.3/s Avg: 0 Min: 0 Max:
68 Err: 10098 (5.66%)

summary + 11216 in 00:00:30 = 373.9/s Avg: 0 Min: 0 Max:
20 Err: 677 (6.04%) Active: 750 Started: 750 Finished: 0

summary = 189752 in 00:09:03 = 349.7/s Avg: 0 Min: 0 Max: 68 Err: 10775 (5.68%)

summary + 11262 in 00:00:30 = 375.4/s Avg: 0 Min: 0 Max: 21 Err: 662 (5.88%) Active: 750 Started: 750 Finished: 0

summary = 201014 in 00:09:33 = 351.1/s Avg: 0 Min: 0 Max: 68 Err: 11437 (5.69%)

summary + 11260 in 00:00:30 = 375.3/s Avg: 0 Min: 0 Max: 20 Err: 672 (5.97%) Active: 743 Started: 750 Finished: 7

summary = 212274 in 00:10:03 = 352.3/s Avg: 0 Min: 0 Max: 68 Err: 12109 (5.70%)

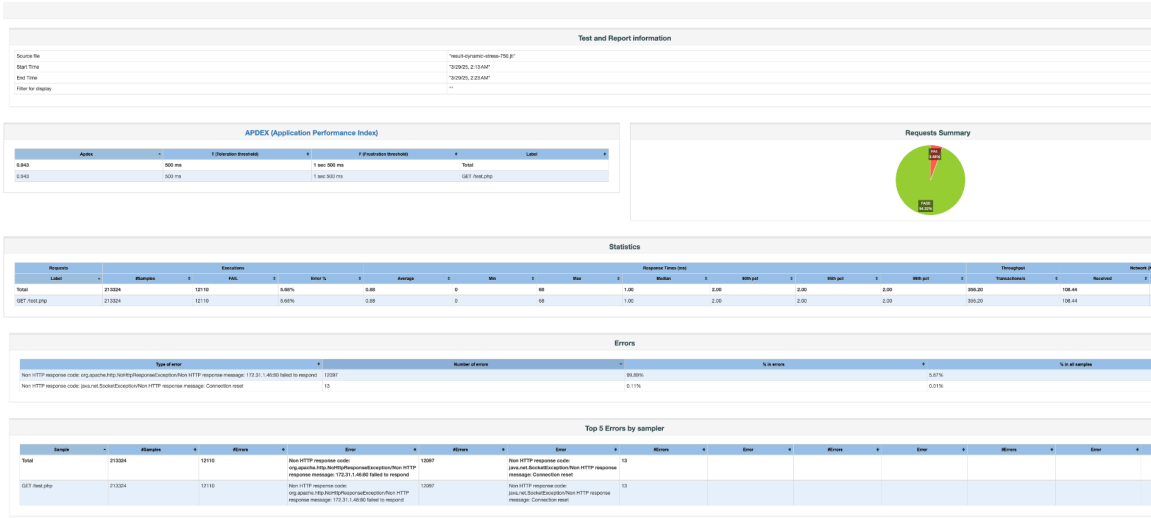
summary + 1050 in 00:00:03 = 350.4/s Avg: 0 Min: 0 Max: 3 Err: 1 (0.10%) Active: 0 Started: 750 Finished: 750

summary = 213324 in 00:10:06 = 352.3/s Avg: 0 Min: 0 Max: 68 Err: 12110 (5.68%)

Tidying up ... @ 2025 Mar 29 02:23:33 UTC (1743215013000)

... end of run

ii. View result: python3 -m http.server 8080
Go to: http://54.176.203.205:8080/



f. Analyze
The stress test for 750 dynamic PHP users shows a clear performance boundary being reached. While the 700-user test ran flawlessly with 0% errors, the 750-user test resulted in a 5.68% error rate, primarily due to non-HTTP response errors (i.e., server not responding). This indicates that the server struggled to keep up with the increased load beyond 700 users. The throughput remained high at ~352 transactions/sec, and response times were consistent, but the rising number of failed executions suggests saturation. Based on these findings, we can conclude that t**he stable upper boundary for dynamic PHP requests on the current setup is around 700 concurrent users**. Beyond this, system reliability begins to degrade.

4 Playload Size Limit Stress Testing:

1. 20MB + 500 Users

a. Set up file
dd if=/dev/urandom of=/var/www/html/test-20MB.txt bs=1M count=20

b. Verify the file

```
root@ip-172-31-1-46:~# curl -I http://localhost/test-20MB.txt
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Date: Sat, 29 Mar 2025 02:57:41 GMT
Content-Type: text/plain
Content-Length: 20971520
Last-Modified: Sat, 29 Mar 2025 02:57:31 GMT
Connection: keep-alive
ETag: "67e7619b-1400000"
Accept-Ranges: bytes
```

c. Configurations

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Payload Stress Test Plan (500 Users)" enabled="true">
      <stringProp name="TestPlan.comments">Stress test 20MB payload via GET</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
```

```
    <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
      <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
          <stringProp name="Argument.name">HOST</stringProp>
          <stringProp name="Argument.value">52.53.164.74</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
          <stringProp name="Argument.name">PORT</stringProp>
          <stringProp name="Argument.value">80</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
          <stringProp name="Argument.name">PROTOCOL</stringProp>
          <stringProp name="Argument.value">http</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
      </collectionProp>
    </elementProp>
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
  <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Payload Stress Group (500 Users)" enabled="true">
      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">500</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /test-20MB.txt" enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
```

```
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.path">/test-20MB.txt</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.connect_timeout">10000</stringProp>
        <stringProp name="HTTPSampler.response_timeout">60000</stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK" enabled="true">
          <collectionProp name="Assertion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Random Delay Timer"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
      </UniformRandomTimer>
      <hashTree/>
    </hashTree>
  </hashTree>
 </hashTree>
</jmeterTestPlan>
```

d. Run test
   jmeter -n -t test-playload-stress-20MB.jmx -l test-playload-stress-20MB.jtl -e -o report-20MB
e. View result

root@ip-172-31-15-56:~/test-0328# jmeter -n -t test-playload-stress-20MB.jmx -l test-playload-stress-20MB.jtl -e -o report-20MB WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future releaseB WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release Creating summariser <summary> Created the tree successfully using test-playload-stress-20MB.jmx Starting standalone test @ 2025 Mar 29 03:03:25 UTC (1743217405519) Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445 summary + 1 in 00:00:07 = 0.2/s Avg: 272 Min: 272 Max: 272 Err: 0 (0.00%) Active: 14 Started: 14 Finished: 0 java.lang.OutOfMemoryError: Java heap space Dumping heap to java_pid18771.hprof ... Killed

f. Analyze the error:
The previous `java.lang.OutOfMemoryError: Java heap space` occurred because JMeter attempted to load and store the full response bodies of each 20MB file requested by 700 virtual users. Since each user request resulted in a large payload, the cumulative memory consumption quickly overwhelmed the JVM's default heap size, especially with response saving enabled for all results. JMeter is not inherently optimized for large binary payload testing out of the box, so without adjusting memory limits or disabling result storage, the heap fills up rapidly and the process crashes.

g. What's next
due to the large 20MB payload size, the JMeter client consistently ran into `OutOfMemoryError` at 500 users. This is because JMeter's default Java heap space is typically limited to 512MB and handling multiple large responses in parallel quickly exhausted available memory. As a result, we progressively scaled down and ultimately tested starting with 20 users to ensure the test could run to completion without exhausting the system's memory.

2. 20 MB + 20 users

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Payload Stress Test Plan (10 Users)" enabled="true">
      <stringProp name="TestPlan.comments">Stress test 20MB payload via GET</stringProp>
```

```
    <boolProp name="TestPlan.functional_mode">false</boolProp>
    <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
    <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
      <collectionProp name="Arguments.arguments">
        <elementProp name="HOST" elementType="Argument">
          <stringProp name="Argument.name">HOST</stringProp>
          <stringProp name="Argument.value">52.53.164.74</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PORT" elementType="Argument">
          <stringProp name="Argument.name">PORT</stringProp>
          <stringProp name="Argument.value">80</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
        <elementProp name="PROTOCOL" elementType="Argument">
          <stringProp name="Argument.name">PROTOCOL</stringProp>
          <stringProp name="Argument.value">http</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
        </elementProp>
      </collectionProp>
    </elementProp>
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
  <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Payload Stress Group (10 Users)" enabled="true">
      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <stringProp name="LoopController.loops">500</stringProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">20</stringProp>
      <stringProp name="ThreadGroup.ramp_time">60</stringProp>
      <boolProp name="ThreadGroup.scheduler">true</boolProp>
      <stringProp name="ThreadGroup.duration">600</stringProp>
      <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
      <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /test-20MB.txt" enabled="true">
        <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
```

```
          <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
        <stringProp name="HTTPSampler.port">${PORT}</stringProp>
        <stringProp name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
        <stringProp name="HTTPSampler.path">/test-20MB.txt</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.connect_timeout">10000</stringProp>
        <stringProp name="HTTPSampler.response_timeout">60000</stringProp>
      </HTTPSamplerProxy>
      <hashTree>
        <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK" enabled="true">
          <collectionProp name="Assertion.test_strings">
            <stringProp name="49586">200</stringProp>
          </collectionProp>
          <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
      </hashTree>
      <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Random Delay Timer"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
      </UniformRandomTimer>
      <hashTree/>
    </hashTree>
  </hashTree>
 </hashTree>
</jmeterTestPlan>
```

a. Run the test
   jmeter -n -t test-playload-stress-20MB-20users.jmx -l
   test-playload-stress-20MB-20users.jtl -e -o report-20MB-20users

b. Result

```
root@ip-172-31-15-56:~/playload-stresstest# jmeter -n -t
test-playload-stress-20MB-20users.jmx -l test-playload-stress-20MB-20users.jtl
WARN StatusConsoleListener The use of package scanning to locate plugins is
deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is
deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is
deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is
deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using test-playload-stress-20MB-20users.jmx
Starting standalone test @ 2025 Mar 29 04:24:05 UTC (1743222245519)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message
on port 4445
summary +     29 in 00:00:24 =    1.2/s Avg:   269 Min:   183 Max:   470 Err:     0
(0.00%) Active: 7 Started: 7 Finished: 0
java.lang.OutOfMemoryError: Java heap space
Dumping heap to java_pid23732.hprof ...
Heap dump file created [502522547 bytes in 2.792 secs]
Uncaught Exception java.lang.OutOfMemoryError: Java heap space in thread
Thread[#29,Payload Stress Group (50 Users) 1-3,5,main]. See log file for details.
summary +     99 in 00:00:32 =    3.1/s Avg:  1075 Min:   188 Max:  5159 Err:     0
(0.00%) Active: 17 Started: 18 Finished: 1
summary =    128 in 00:00:56 =    2.3/s Avg:   892 Min:   183 Max:  5159 Err:     0
(0.00%)
Uncaught Exception java.lang.OutOfMemoryError: Java heap space in thread
Thread[#35,Payload Stress Group (50 Users) 1-9,5,main]. See log file for details.
Uncaught Exception java.lang.OutOfMemoryError: Java heap space in thread
Thread[#40,Payload Stress Group (50 Users) 1-14,5,main]. See log file for
details.
Uncaught Exception java.lang.OutOfMemoryError: Java heap space in thread
Thread[#38,Payload Stress Group (50 Users) 1-12,5,main]. See log file for
details.
```

c. Analyze

The 20-user payload stress test with 20MB files resulted in a Java heap space OutOfMemoryError, indicating that the testing machine (not the server) could not allocate enough memory to handle the load generated by concurrent large payloads. While the test initially progressed with stable throughput and response times under 6 seconds, the growing memory demand led to critical failures

shortly after ramp-up. This confirms that, under the current hardware limitations (1 vCPU, 1GB RAM), 20 concurrent users requesting large files exceeds the testing environment's capacity. To ensure meaningful and stable results, we will proceed with testing using 10 users, which has been verified to complete successfully without memory-related errors.

3. 20MB + 10 Users

a. Configurations

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Payload
Stress Test Plan (10 Users)" enabled="true">
      <stringProp name="TestPlan.comments">Stress test 20MB payload via
GET</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments">
        <collectionProp name="Arguments.arguments">
          <elementProp name="HOST" elementType="Argument">
            <stringProp name="Argument.name">HOST</stringProp>
            <stringProp name="Argument.value">52.53.164.74</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PORT" elementType="Argument">
            <stringProp name="Argument.name">PORT</stringProp>
            <stringProp name="Argument.value">80</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="PROTOCOL" elementType="Argument">
            <stringProp name="Argument.name">PROTOCOL</stringProp>
            <stringProp name="Argument.value">http</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
        </collectionProp>
      </elementProp>
```

```
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
   </TestPlan>
   <hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Payload Stress Group (10 Users)" enabled="true">
     <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
     <elementProp name="ThreadGroup.main_controller"
elementType="LoopController">
      <boolProp name="LoopController.continue_forever">false</boolProp>
      <stringProp name="LoopController.loops">500</stringProp>
     </elementProp>
     <stringProp name="ThreadGroup.num_threads">10</stringProp>
     <stringProp name="ThreadGroup.ramp_time">60</stringProp>
     <boolProp name="ThreadGroup.scheduler">true</boolProp>
     <stringProp name="ThreadGroup.duration">600</stringProp>
     <stringProp name="ThreadGroup.delay">5</stringProp>
    </ThreadGroup>
    <hashTree>
     <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="GET /test-20MB.txt" enabled="true">
      <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
       <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <stringProp name="HTTPSampler.domain">${HOST}</stringProp>
      <stringProp name="HTTPSampler.port">${PORT}</stringProp>
      <stringProp name="HTTPSampler.protocol">${PROTOCOL}</stringProp>
      <stringProp name="HTTPSampler.path">/test-20MB.txt</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp
name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.connect_timeout">10000</stringProp>
      <stringProp name="HTTPSampler.response_timeout">60000</stringProp>
     </HTTPSamplerProxy>
     <hashTree>
      <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="Assert 200 OK" enabled="true">
       <collectionProp name="Assertion.test_strings">
        <stringProp name="49586">200</stringProp>
       </collectionProp>
```

```
          <stringProp
name="Assertion.test_field">Assertion.response_code</stringProp>
          <intProp name="Assertion.test_type">8</intProp>
        </ResponseAssertion>
        <hashTree/>
       </hashTree>
       <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Random Delay Timer"
enabled="true">
        <stringProp name="ConstantTimer.delay">1000</stringProp>
        <stringProp name="RandomTimer.range">2000</stringProp>
       </UniformRandomTimer>
       <hashTree/>
      </hashTree>
    </hashTree>
  </hashTree>
</jmeterTestPlan>
```

b. Run the test
   jmeter -n -t test-playload-stress-20MB-10users.jmx -l
   test-playload-stress-20MB-10users.jtl -e -o report-20MB-10users
c. Result
   i.

```
root@ip-172-31-15-56:~/playload-stresstest# jmeter -n -t
test-playload-stress-20MB-10users.jmx -l
test-playload-stress-20MB-10users.jtl -e -o report-20MB-10users
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate
plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using
test-playload-stress-20MB-10users.jmx
Starting standalone test @ 2025 Mar 29 04:08:44 UTC (1743221324143)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump
message on port 4445
summary +      7 in 00:00:16 =    0.4/s Avg:   215 Min:   194 Max:   301
Err:     0 (0.00%) Active: 2 Started: 2 Finished: 0
summary +     59 in 00:00:30 =    2.0/s Avg:   281 Min:   183 Max:   590
Err:     0 (0.00%) Active: 7 Started: 7 Finished: 0
```

summary = 66 in 00:00:46 = 1.4/s Avg: 274 Min: 183 Max: 590 Err: 0 (0.00%)
summary + 109 in 00:00:30 = 3.6/s Avg: 538 Min: 192 Max: 1199 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 175 in 00:01:16 = 2.3/s Avg: 438 Min: 183 Max: 1199 Err: 0 (0.00%)
summary + 119 in 00:00:30 = 4.0/s Avg: 680 Min: 192 Max: 1526 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 294 in 00:01:46 = 2.8/s Avg: 536 Min: 183 Max: 1526 Err: 0 (0.00%)
summary + 119 in 00:00:30 = 3.9/s Avg: 521 Min: 202 Max: 1120 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 413 in 00:02:16 = 3.0/s Avg: 532 Min: 183 Max: 1526 Err: 0 (0.00%)
summary + 89 in 00:00:31 = 2.9/s Avg: 573 Min: 195 Max: 9646 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 502 in 00:02:47 = 3.0/s Avg: 539 Min: 183 Max: 9646 Err: 0 (0.00%)
summary + 11 in 00:00:29 = 0.4/s Avg: 22350 Min: 3233 Max: 34468 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 513 in 00:03:16 = 2.6/s Avg: 1007 Min: 183 Max: 34468 Err: 0 (0.00%)
summary + 15 in 00:00:30 = 0.5/s Avg: 16572 Min: 3118 Max: 28412 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 528 in 00:03:47 = 2.3/s Avg: 1449 Min: 183 Max: 34468 Err: 0 (0.00%)
summary + 11 in 00:00:29 = 0.4/s Avg: 24801 Min: 17424 Max: 29636 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 539 in 00:04:16 = 2.1/s Avg: 1926 Min: 183 Max: 34468 Err: 0 (0.00%)
summary + 11 in 00:00:30 = 0.4/s Avg: 24170 Min: 12900 Max: 33695 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 550 in 00:04:46 = 1.9/s Avg: 2371 Min: 183 Max: 34468 Err: 0 (0.00%)
summary + 12 in 00:00:35 = 0.3/s Avg: 25324 Min: 11920 Max: 34613 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 562 in 00:05:21 = 1.8/s Avg: 2861 Min: 183 Max: 34613 Err: 0 (0.00%)
summary + 10 in 00:00:25 = 0.4/s Avg: 23719 Min: 12747 Max: 39989 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0
summary = 572 in 00:05:46 = 1.7/s Avg: 3225 Min: 183 Max: 39989 Err: 0 (0.00%)
summary + 13 in 00:00:30 = 0.4/s Avg: 28070 Min: 15571 Max: 46317 Err: 0 (0.00%) Active: 10 Started: 10 Finished: 0

```
summary =    585 in 00:06:15 =    1.6/s Avg:  3778 Min:   183 Max: 46317
Err:     0 (0.00%)
summary +     10 in 00:00:31 =    0.3/s Avg: 22791 Min: 13605 Max:
38639 Err:     0 (0.00%) Active: 10 Started: 10 Finished: 0
summary =    595 in 00:06:46 =    1.5/s Avg:  4097 Min:   183 Max: 46317
Err:     0 (0.00%)
summary +     10 in 00:00:30 =    0.3/s Avg: 26714 Min: 14512 Max:
41819 Err:     0 (0.00%) Active: 10 Started: 10 Finished: 0
summary =    605 in 00:07:17 =    1.4/s Avg:  4471 Min:   183 Max: 46317
Err:     0 (0.00%)
summary +     12 in 00:00:30 =    0.4/s Avg: 25070 Min: 17139 Max:
38118 Err:     0 (0.00%) Active: 10 Started: 10 Finished: 0
summary =    617 in 00:07:47 =    1.3/s Avg:  4872 Min:   183 Max: 46317
Err:     0 (0.00%)
summary +     11 in 00:00:32 =    0.3/s Avg: 26274 Min: 16429 Max:
48100 Err:     0 (0.00%) Active: 10 Started: 10 Finished: 0
summary =    628 in 00:08:19 =    1.3/s Avg:  5246 Min:   183 Max: 48100
Err:     0 (0.00%)
summary +     21 in 00:00:56 =    0.4/s Avg: 24956 Min: 15641 Max:
35954 Err:     0 (0.00%) Active: 10 Started: 10 Finished: 0
summary =    649 in 00:09:16 =    1.2/s Avg:  5884 Min:   183 Max: 48100
Err:     0 (0.00%)
summary +     12 in 00:00:34 =    0.4/s Avg: 24121 Min: 16383 Max:
36535 Err:     0 (0.00%) Active: 10 Started: 10 Finished: 0
summary =    661 in 00:09:49 =    1.1/s Avg:  6215 Min:   183 Max: 48100
Err:     0 (0.00%)
summary +     13 in 00:00:25 =    0.5/s Avg: 24307 Min: 13692 Max:
47215 Err:     0 (0.00%) Active: 0 Started: 10 Finished: 10
summary =    674 in 00:10:14 =    1.1/s Avg:  6564 Min:   183 Max: 48100
Err:     0 (0.00%)
Tidying up ...    @ 2025 Mar 29 04:18:58 UTC (1743221938923)
... end of run
```

d. Analyze

   For the payload size stress test, we configured 10 users to repeatedly request a 20MB static file. The test completed successfully without any errors, showing that the server could handle high payload transfers with low concurrency. However, as response times climbed above 40 seconds for some requests, this confirms that while 10 users is stable, the large file size puts significant strain on server response time and throughput under limited hardware (1 vCPU, 1GB RAM). This highlights the trade-off between payload size and concurrency in resource-constrained environments.