Debugging Tests in BlazeMeter

When you encounter issues where you feel that you are not getting the load or the response time that you are expecting, you will need to look at two things. One is the log and the other is the monitoring report. Let's learn about some basic steps to debug your tests in BlazeMeter

First Step

When you think there is an issue with your test, the first thing you do is to ensure that the test runs locally.

Second Step

When you encounter issues where you feel that you are not getting the load or the response time that you are expecting, you will need to look at two things. One is the log and the other is the monitoring report.

Common Example for Using Logs

Consider a scenario where BlazeMeter is unable to read the CSV files or the certificate is not loaded onto the key store, or you tried to upload a plugin which is not supported, the logs will be the first place to check.

Common Example for Using Console Log

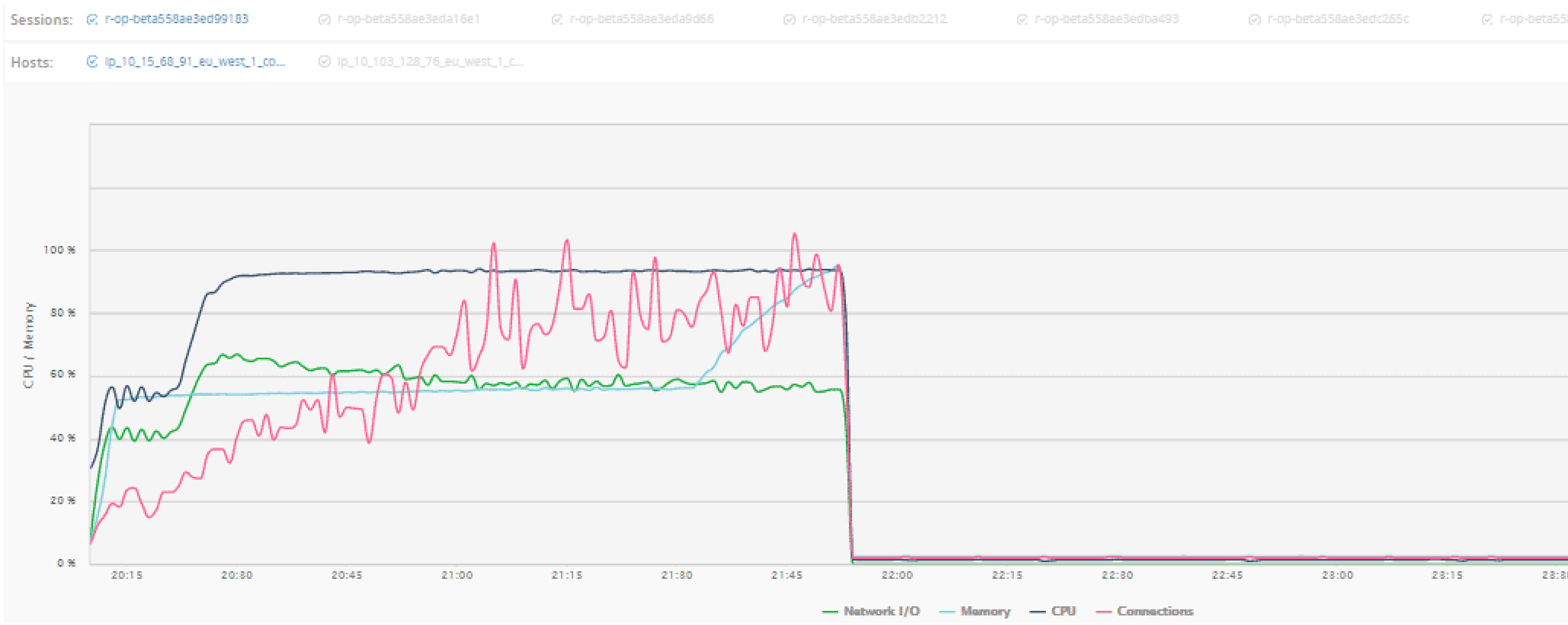
In the console log, you can see issues such as not being able to connect to remote host, configuration issues, and in the engine log, you may see issues where you were unable to load the CSV file.

Key Thing to Identify in Logs

Another important thing to look for in the logs is when the threads started and died. A very common mistake is setting a CSV Data Set Config element to “Stop thread on EOF” (end of file). Then you run a thread and wonder why are you not getting the maximum load that you aimed for. You can go to the logs and then you see that threads are starting and then JMeter starts terminating threads so it is either an error or a wrong configuration from the script.

Checking Engine Health

If everything looks fine in the logs, then you go to the engine health. There are chances that you might have just crashed the engine or exhausted the CPU, memory, or throughput. If the Engine Health is up to 70-75% then you are fine. If you are over 80% then, you should stop and reduce the number of threads per engine.

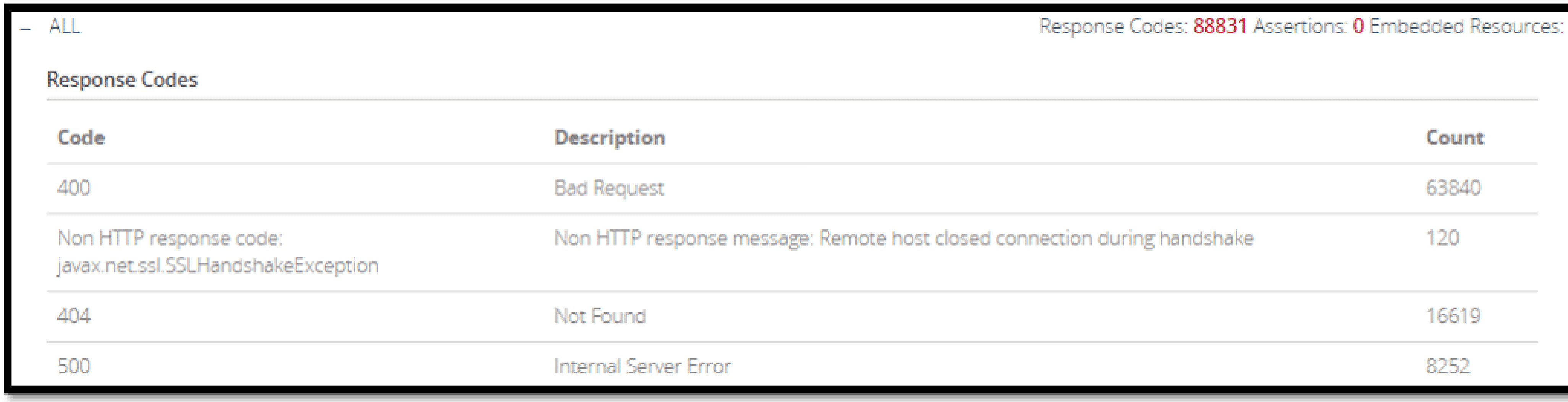


Engine Health Report

The Engine Health Report allows us to ensure that the engines of BlazeMeter are not the bottleneck.

Please note that CPU should be lower than 75% and Memory should be lower than 85% and Network < 300Mb/s. If these numbers are exceeded, you should spread your threads across a larger number of engines.

If you see the CPU rising and suddenly dropping to zero, for the entire time of the test, that means JMeter died due to some issue.



– ALL Response Codes: 88831 Assertions: 0 Embedded Resources:

Response Codes

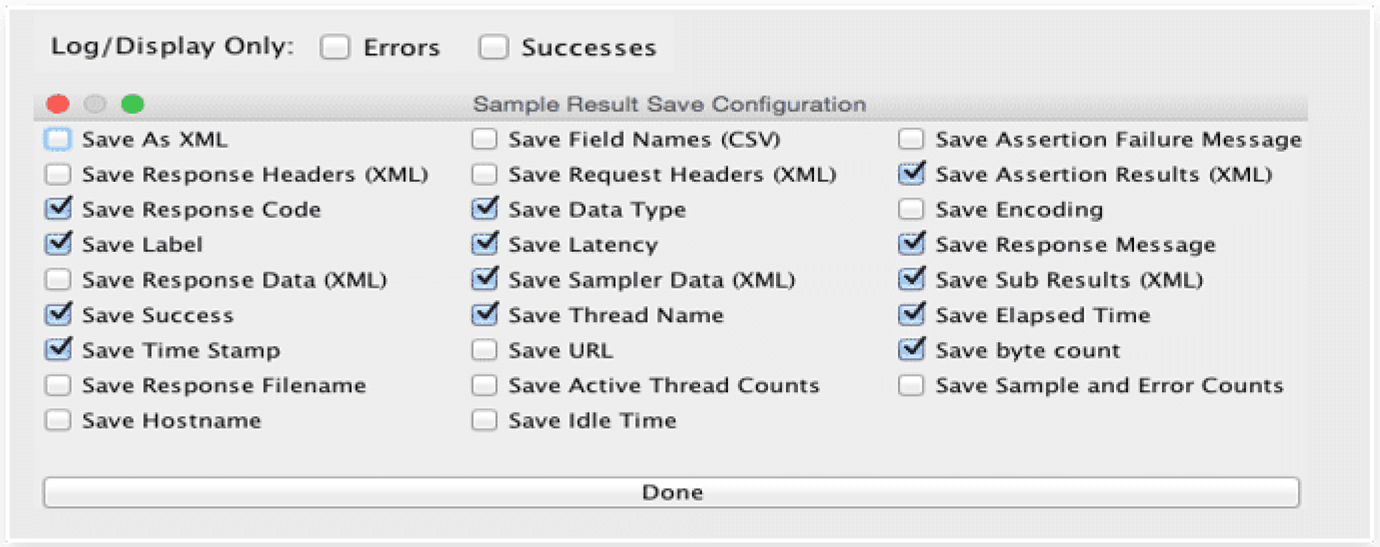
Code	Description	Count
400	Bad Request	63840
Non HTTP response code: javax.net.ssl.SSLHandshakeException	Non HTTP response message: Remote host closed connection during handshake	120
404	Not Found	16619
500	Internal Server Error	8252

Using the Errors Report

The Errors tab, as it's name suggests, contains the errors that were returned by the web-server under the test as a result of HTTP request. We can see all errors received during the test run, categorized by Labels (pages) and Error types.

Using the JTLs and More File

At the end of every test, you get the jtls_and_more.zip file. This ZIP file includes the JMX file that you've uploaded which has been modified by BlazeMeter while running, The JTL file which contains the results of the test run, CSVs and any additional file you might have used for that test run. Please note that, if you have terminated the test instead of Shutting it down gracefully, or perhaps the test ended prematurely in an unexpected manner, the jtls_and_more.zip will not be generated.



Log/Display Only: [ ] Errors [ ] Successes

Sample Result Save Configuration

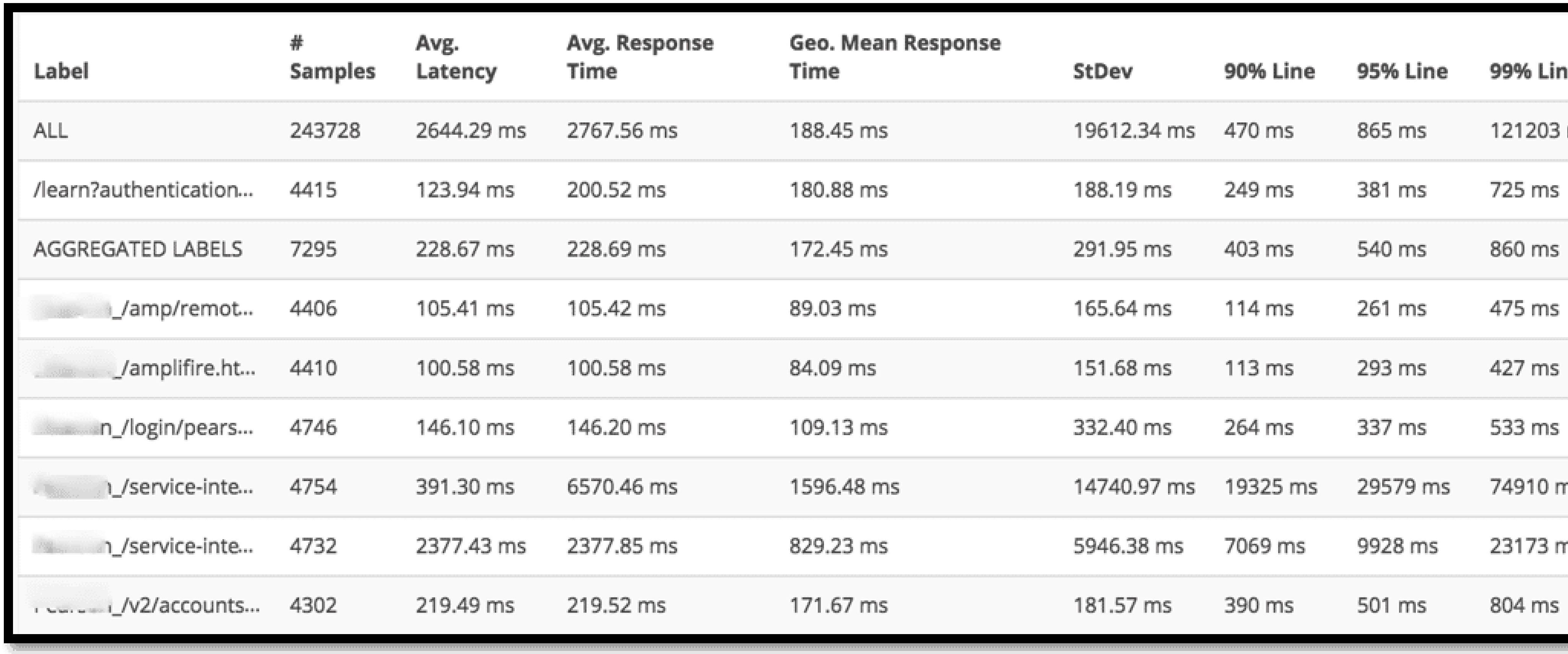
[ ] Save As XML	[ ] Save Field Names (CSV)	[ ] Save Assertion Failure Message
[ ] Save Response Headers (XML)	[ ] Save Request Headers (XML)	[✓] Save Assertion Results (XML)
[✓] Save Response Code	[✓] Save Data Type	[ ] Save Encoding
[✓] Save Label	[✓] Save Latency	[✓] Save Response Message
[ ] Save Response Data (XML)	[✓] Save Sampler Data (XML)	[✓] Save Sub Results (XML)
[✓] Save Success	[✓] Save Thread Name	[✓] Save Elapsed Time
[✓] Save Time Stamp	[ ] Save URL	[✓] Save byte count
[ ] Save Response Filename	[ ] Save Active Thread Counts	[ ] Save Sample and Error Counts
[ ] Save Hostname	[ ] Save Idle Time	

Adding Listeners

In case you want more information that what is available in the JTL file, you can add a listener, which allows you to see all the requests and responses and all other details. JMeter will create a file based on the filename that you gave and it will be a part of the JTL.zip. You can run JMeter with a listener and download the sample JTL and get all the details. After you have made changes to your script, ensure that you disable listeners before finally executing the script.

Using Request Stats Report to Break Down Scenarios

You can use Request Stats report to identify specific requests and transactions which have issues in them. You can specify which columns you want to see in this report. One key thing you can do is reorder the report based on the error percentage or error count and identify the labels that created most of the errors. Then, we can filter down to the branches of the test that created most errors during the load test and effectively debug the test.

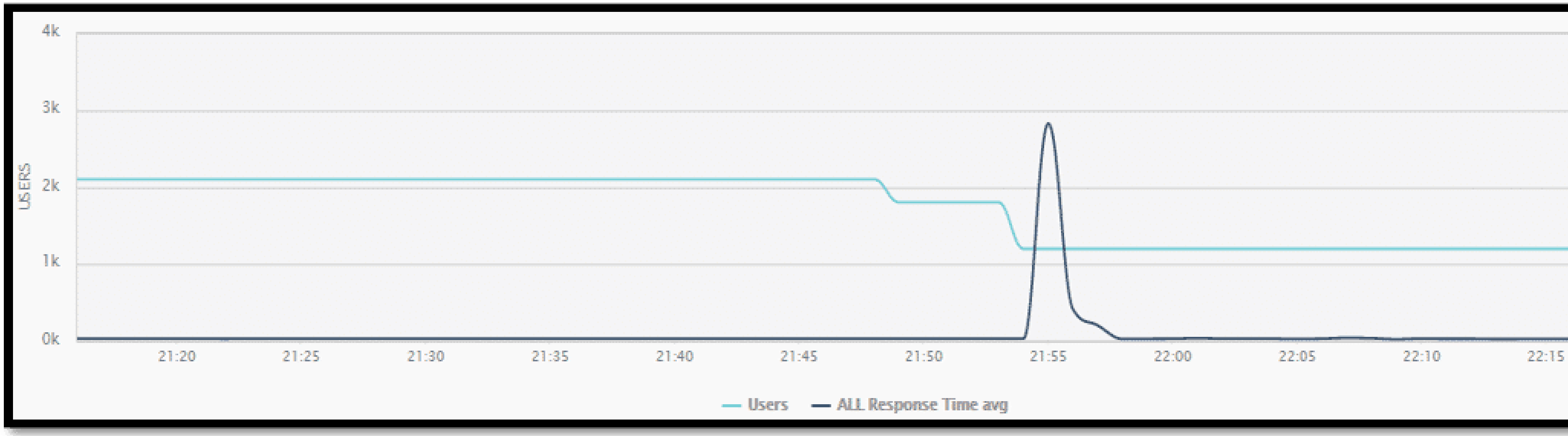


Label	# Samples	Avg. Latency	Avg. Response Time	Geo. Mean Response Time	StDev	90% Line	95% Line	99% Lin
ALL	243728	2644.29 ms	2767.56 ms	188.45 ms	19612.34 ms	470 ms	865 ms	121203
/learn?authentication...	4415	123.94 ms	200.52 ms	180.88 ms	188.19 ms	249 ms	381 ms	725 ms
AGGREGATED LABELS	7295	228.67 ms	228.69 ms	172.45 ms	291.95 ms	403 ms	540 ms	860 ms
[redacted]_/amp/remot...	4406	105.41 ms	105.42 ms	89.03 ms	165.64 ms	114 ms	261 ms	475 ms
[redacted]_/amplifire.ht...	4410	100.58 ms	100.58 ms	84.09 ms	151.68 ms	113 ms	293 ms	427 ms
[redacted]n_/login/pears...	4746	146.10 ms	146.20 ms	109.13 ms	332.40 ms	264 ms	337 ms	533 ms
[redacted]_/service-inte...	4754	391.30 ms	6570.46 ms	1596.48 ms	14740.97 ms	19325 ms	29579 ms	74910 m
[redacted]_/service-inte...	4732	2377.43 ms	2377.85 ms	829.23 ms	5946.38 ms	7069 ms	9928 ms	23173 m
[redacted]_/v2/accounts...	4302	219.49 ms	219.52 ms	171.67 ms	181.57 ms	390 ms	501 ms	804 ms

Example of Engine Crash

–

When you see a report like this where the response time has spiked and the number of users has come down then one conclusion you can usually make is that one of your engines has crashed.


4k
3k
2k
1k
0k
USERS
21:20
21:25
21:30
21:35
21:40
21:45
21:50
21:55
22:00
22:05
22:10
22:15
Users
ALL Response Time avg