Calibrating Tests in BlazeMeter

Test calibration is important and helps you to determine the number of threads per engine and number of engines per console. This way you can verify when you run a test that your CPU consumption does not go over 75%. If you don’t perform test calibration, you might use too many threads per engine or too many engines per console which will spike up your CPU consumption to reach the critical level (i.e. 90%) and your test may stop abruptly.

Here are some key activities you need to perform to calibrate tests in CA BlazeMeter.

Simulate Browser’s Functionality

Add ‘Cache manager’ to the script, Add ‘Cookie manager’ to the script, Add ‘DNS Cache Manager’ to the script

Add Delays

Add a reasonable delay between threads and use Uniform, Gaussian or Constant timers

Disable Listeners

Please disable\remove all listeners from your script as they are not used in BlazeMeter because they consume a lot of resources. It's fine to occasionally use log listeners.

Ramp Up Configuration

It is recommended to use a gradual rampup in order to identify bottlenecks, or load engine limits

Given below are some of the activities you need to perform to validate scripts.

Write Your Script

You can write your script by using the JMeter HTTP(S) Test Script Recorder. You can also use the Chrome Extension to record the script or write the script manually by using JMeter.

Testing Locally with JMeter

Run one thread, one iteration, and use the 'View results Tree' listener. Keep the Log Viewer open. If successful, run with 10-20 threads, loop ‘forever’ for 10 minutes. Look for errors and check stats to ensure the test is running successfully.

Sandbox Testing

The SandBox configuration allows you to test your script, backend, and ensure everything works well. Make sure all assets are uploaded, your environment is open to the BlazeMeter IPs, and remove paths from CSV. Then look for errors and monitor the console’s health.

Let's now learn how to setup the number of users per engine.

Configure BlazeMeter Test

You can start by configuring a BlazeMeter test with the following details.

Threads: 500,Ramp-up: 40 min, Iterations: forever, Duration: 50 min, 1 console, 1 engine

Monitor Health

Run the test for the full length of your final test. While the test is running, check the Engine Health Report. Verify that none of the engines pass the 75% CPU, 85% Memory limit. Locate your console’s tab (to find it, go to the Logs Report console’s log and look for its private IP). This should not reach the 75% CPU or 85% Memory limit.

Re-configuring the BlazeMeter Test

You can now increase the number of threads to 700. The updated configuration of the test is given below.

Threads: 700, Ramp-up: 40 min, Iterations: forever, Duration: 50 min, 1 console, 1 engine.

Repeat this process till you reach 1000 threads.

Drawing Conclusions

You should be checking to see when you hit 70% CPU Utilization or 80% Memory usage. Notice how many users you loaded until that point. Run the test again with this amount and verify the engine is not overwhelmed.

Let's now learn how to configure the number of engines per console.

Configure BlazeMeter Test

Use the same test that was calibrated for one engine and change only the number of engines by raising them to 14.

Monitor Health

Run the test for the full length of your final test. While the test is running, check the Engine Health Report. Verify that none of the engines pass the 75% CPU, 85% Memory limit. Locate your console’s tab (to find it, go to the Logs Report console’s log and look for its private IP). This should not reach the 75% CPU or 85% Memory limit.

Re-configuring the BlazeMeter Test

If the console reached those limits - decrease the number of engines and run again until the console is within these limits. By the end of this step, you should know the Users per Cluster you'll have and the Hits/s per Cluster you'll reach.