Apache JMeter is an awesome tool that provides excellent scripting capabilities. You can simulate almost anything using JMeter scripting like log in, posting of forms, various browsers and so on but executing tests using JMeter is a totally different matter. Apache JMeter is an open source load testing tool used to perform load test on web applications (client/server). JMeter is not a browser rather it’s purely a Java based application that works at the protocol level.

In order to check the performance of any web application, you need to first create a test plan consisting of the following elements:

✳ Thread Group

✳ Controllers

✳ Samplers

✳ Logic controllers

✳ Test Fragments

✳ Listeners

✳ Timers

✳ Assertions

✳ Configuration elements

✳ Pre-processor elements

✳ Post-processor elements

After creating the test plan, you need to create the test scripts and run those scripts.

JMeter will execute the test plan elements in the following order:

1. Configuration elements

2. Pre-Processors

3. Timers

4. Sampler

5. Post-Processors (unless SampleResult is null)

6. Assertions (unless SampleResult is null)

7. Listeners (unless SampleResult is null)

Before starting Performance Testing using JMeter, below are the common questions that one should clarify with the customer.

1. How many concurrent users do we expect the application to support on an average at any given point in time?

2. How many concurrent users do we expect the application to support on an average during peak duration?

3. List of pages/activities that would be used more often to prioritize for Performance testing?

4. What is the acceptable response time for the pages for performance testing?

5. What will be the duration? How long will the load remain on the system?

6. What will be the Infrastructure set up and application architecture?

7. How will Server Monitoring be handled?

8. Is there any scalability requirement over the period of time ?

It would be really easy to prepare a performance Test Plan once we get all answers to above questions. Once Performance Test Plan is prepared, next step is to develop scripts.

Best Practices for JMeter Script Preparation

There are number of tutorials available over the internet which gives us step by step guide for developing scripts in JMeter. Develop the jmx files according to the project requirement. Below are some standards that we should follow while developing scripts:

✳ Remove unnecessary requests from the plan. (Lets say if you are recording with HTTP(S) Test Script Recorder from Non-Test Elements)

Organize the samplers and other elements in most easy way.

✳ Follow Naming convention for all elements.

✳ Always keep “Retrieve all embedded resources” checked to make it behave more like a browser.

✳ Add listeners appropriately. Keep Result Tree disable when run for large number of users.

✳ Graphs use more memory. Keep it disabled, if not required.

✳ Try to run the script in non-GUI mode for simulating large concurrency.

✳ Only save the data that you need.

✳ Make sure how to use cookie manager and cache manager along with your requirement.

✳ Do not forget to add timers as per your requirement.

✳ Always use relative path for csv file if used in the script.

✳ Use multiple instances of JMeter in case the large number of threads/users is required to be simulated.

Common Issues

The Console Freezes

The JMeter console running in GUI mode can consume a lot of CPU and memory. Memory consuming listeners such as “View Results Tree” or any of the graphical reporting listeners can cause the console to freeze without providing any prior alert. The GUI console is excellent for running a small load and developing and/or debugging your script. We don't recommend running high capacity loads using the GUI console as in most cases, it will freeze halfway through the test.

Symptom is the JMeter GUI console will stop responding

What not to do

1. Do not use the GUI console for large capacity load.

2. Do not have graphical reporting listeners enabled while running.

3. Do not have memory consuming listeners enabled while running.

What to do

1. Use the GUI console for script development and debugging only.

2. Use the GUI console for small loads (<50 simultaneous users).

3. Run JMeter headless for any load over 50 simultaneous users.

The Console Doesn’t Collect Engine Data (it’s stopped responding again) - Using Distributed Architecture

When running a test using a JMeter distributed architecture that might include, (for example) 4 engines all the data that each engine generates is tunneled through the JMeter console. That is A LOT of data. This means that the console needs to deal with all data gathered from all the JMeter engines - in parallel. This can be an extremely heavy burden for the JMeter console.

Most likely, 2 things will happen

1. 100% CPU utilization

2. Out of memory exception

In both scenarios the console will freeze and will stop reacting to the engines.

Symptoms

If running in GUI mode - The GUI console will stop responding.

ConnectException: Connection refused to host: 10.10.110.110

This means that the connection was refused by the console.

What not to do

1. Do not run in GUI mode.

2. Do not over-stress your testing environment

What to do

1. Run headless.

2. Monitor the console vital signs (i.e. CPU and memory consumption).

3. See that the console is responding (for example that it continues to write to a JTL file).

4. Assuming you use strong resources for your test, keep the maximum hits per second lower than 1000 per second.

The Engine is Running Out of Memory

When running a test using a JMeter distributed architecture an engine can often dump its core or produce an out of memory exception. It can also freeze before being able to write this exception to the log. This is usually caused due to a too high number of threads or a too intensive test.

Either way, the engine will stop generating traffic and should log the exception to its log file.

Symptom

If the engine was able to log the exception to its log file, you can find exception like this (example):ERROR - JMeter.threads.JMeterThread: Test failed! java.lang.OutOfMemoryError: Java heap space

What not to do

1. Do not over-stress your testing environment.

What to do

1. Monitor the engines vital signs (i.e. CPU and memory consumption).

2. See that the engine is responding (look at its log file).

3. Test a single engine to see that it can actually sustain the load before running the full blown test. If one engine will run well, it can imply that the others will run well.

4. Keep users between 100-300 for a medium size server.

5. Keep users between 200-600 for a strong server.

6. If an exception is encountered reduce the number of threads.

Happy Testing 🤝🫂