INDEX

JMeter

1. Introduction

- jMeter is an Open Source testing software.
- It is a Java application for performance testing.

1.1. Performance Testing

- Performance Testing is crucial to determine that the web application under test will satisfy high load requirements.
- It can be used to analyze overall server performance under heavy load.
- JMeter Performance Testing includes:
    - Load Testing: Modeling the expected usage by simulating multiple user access the Web services concurrently.
    - Stress Testing: The purpose of the Stress Testing is to find the maximum load the web server can handle.

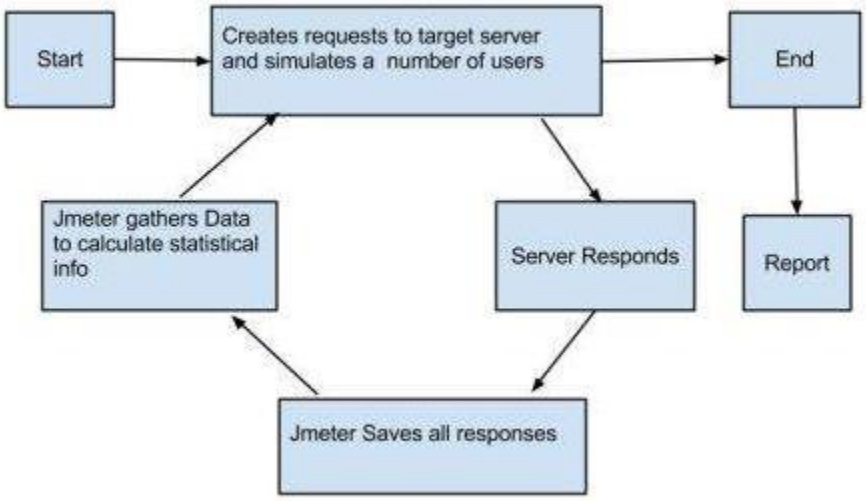
1.2. Load Testing

- Load testing is a kind of Performance Testing which determines a system's performance under real-life load conditions.
- This testing helps determine how the application behaves when multiple users access it simultaneously.
- This testing usually identifies -
    - The maximum operating capacity of an application
    - Determine whether current infrastructure is sufficient to run the application
    - Sustainability of application with respect to peak user load
    - Number of concurrent users that an application can support, and scalability to allow more users to access it.
- Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.

- JMeter can conduct load and performance test for many different server types − Web - HTTP, HTTPS, SOAP, Database via JDBC, LDAP, JMS, Mail - POP3, etc.
- It can also be used to perform automated and functional testing of the applications.

2. jMeter Working

- JMeter simulates a group of users sending requests to a target server, and returns statistics that show the performance/functionality of the target server/application via tables, graphs, etc.



3. jMeter Test Plan

- A Test Plan can be viewed as a container for running tests.
- It defines what to test and how to go about it.
- Elements of Test Plan :-

3.1. Thread Group

- Thread Group elements are the beginning points of your test plan. As the name suggests, the thread group elements control the number of threads JMeter will use during the test.
- The Thread Group Panel holds the following components −
    - **Action to be taken after a Sampler error** −
        - *Continue* to the next element in the test
        - *Stop Thread* to stop the current Thread.
        - *Stop Test* completely, in case you want to inspect the error before it continues running.
    - **Number of Threads** − Simulates the number of users or connections to your server application.
    - **Ramp-Up Period** Defines how long it will take JMeter to get all threads running.
    - **Loop Count** − Defines the number of times to execute the test.
    - **Scheduler Configuration** − You can configure the start and end time of running the test.

3.2. Controllers

- JMeter has two types of Controllers − *Samplers* and *Logic Controllers*.

3.2.1. Samplers

- Samplers allow JMeter to send specific types of requests to a server.
- Some useful samplers are −
    - HTTP Request
    - FTP Request
    - JDBC Request
    - Java Request
    - SOAP/XML Request
    - RPC Requests

3.2.2. Logic Controllers

- Logic Controllers let you control the order of processing of Samplers in a Thread.
- The following list consists of all the Logic Controllers JMeter provides −
    - Simple Controller
    - Loop Controller
    - Once Only Controller
    - Interleave Controller
    - Random Controller
    - Random Order Controller
    - Throughput Controller
    - Runtime Controller
    - If Controller
    - While Controller
    - Switch Controller
    - ForEach Controller
    - Module Controller
    - Include Controller
    - Transaction Controller
    - Recording Controller

3.3. Test Fragments

- This element is purely for code reuse within Test Plans.
- It is not executed unless it is referenced by either a Module-Controller or an Include-Controller.

3.4. Listeners

- Listeners let you view the results of Samplers in the form of tables, graphs, trees, or simple text in some log files.

3.5. Timers

- You can add a timer element which allows you to define a period to wait between each request.

3.6. Assertions

- Assertions allow you to include some validation test on the response of your request made using a Sampler. Using assertions you can prove that your application is returning the correct data.

3.7. Configuration Elements

- Configuration Elements allow you to create defaults and variables to be used by Samplers.
- They are used to add or modify requests made by Samplers.
- They are executed at the start of the scope of which they are part, before any Samplers that are located in the same scope.

3.8. Pre-processor Elements

- A pre-processor element is something that runs just before a sampler executes.

3.9. Post-processor Elements

- A post-processor executes after a sampler finishes its execution.

3.10. Execution Order of Test Elements

- Configuration elements
- Pre-Processors
- Timers
- Sampler
- Post-Processors (unless SampleResult is null)
- Assertions (unless SampleResult is null)
- Listeners (unless SampleResult is null)

4. Running jMeter in non GUI mode

```
jmeter -n -t YourJmeterTestPlan.jmx -l TestResultFile.jtl  -j LogFile.log
```

- -n : Non-GUI mode – This specifies JMeter is to run in non-gui mode
- -t : Name of JMX file that contains Test Plan
- -l : Name of JTL file to capture results to
- -j : Name of Log file to capture execution logs

Output Example

```
Created the tree successfully using test-plans/YourJmeterTestPlan.jmx
Starting the test @ Wed Nov 21 14:56:09 IST 2018 (1542792369706)
Waiting for possible Shutdown/StopTestNow/Heapdump message on port 4445
summary +    317 in 00:00:20 =   16.0/s Avg:   291 Min:   270 Max:   429 Err:     0 (0.00%)
Active: 100 Started: 100 Finished: 0
summary +    565 in 00:00:30 =   18.8/s Avg:   293 Min:   274 Max:   732 Err:     0 (0.00%)
Active: 100 Started: 100 Finished: 0
```

- **317 in 00:00:20 = 16.0/s :** means that in 20 seconds we have sent 317 requests to the server with an average throughput of 16.0 requests per second
- **Avg: 291 -** means the average response time at that moment is 291 milliseconds.
- **Min: 270 -** means the minimum response time from send requests for that period was 270 milliseconds.
- **Max: 429 -** the maximum response time from send requests for that period was 429 milliseconds.
- **Err: 0 (0.00%) -** means we didn't have any errors in requests for that period and the percentage of errors from total requests is 0 accordingly.
- **Active: 100 -** shows the number of active users who were performing requests for this period (here it is 100 active users).
- **Started: 100 -** shows the total number of started threads since the beginning of the tests (here it is 100 threads started).
- **Finished: 0 -** shows the total number of threads that already finished execution since the beginning of tests (here it is 0 threads finished).

5. References

- https://jmeter.apache.org/usermanual/get-started.html
- http://www.testingjournals.com/run-jmeter-command-line-non-gui-mode/
- https://www.blazemeter.com/blog/3-easy-ways-to-monitor-jmeter-non-gui-test-results
- https://www.guru99.com/jmeter-performance-testing.html