**JMETER from APACHE**

# INTRODUCTION

1. Performance Testing: Measuring performance of services such as WebApp/server machine during various loads/tasks being performed.
   * Load Test: carried out to check Maximum number of users/request the server which is hosting WebApp can handle.
   * Stress Test – Analyzing behavior of server when it reaches maximum load.

Basically to measure how system performs under stress.

* + Endurance Test / Soak Test – Involves **testing** a system with a significant load extended over a significant long period of time, to discover how the system behaves under sustained use.
  + Spike Test – This is performed to verify a system's stability during bursts of concurrent user and or systems activity to varying degrees of load over varying time periods.

1. Jmeter is an Open-source Java WebApp used to test/measure **Performance**, **behavior** of elements such as Web Apps, DB, FTP, communication protocol.

i.e., Jmeter simulates a heavy load/users on a server/group of servers/network to test its strength or to analyse overall performance under different load types.

1. Web Services: is a communication service between any 2 devices over **WWW**.
2. API: Application programming interface (API):
   * API is a **set of clearly defined methods of communication** between various components.
   * API makes easier for developers to use certain technologies in building applications, by abstracting the underlying implementation and only exposing objects or actions the developer needs, an API simplifies programming.
3. Types of Web Services:
   * REST (Representational State Transfer) - the primary purpose of this service is to manipulate XML representations of web resources using a uniform set of "stateless" operations.
     1. Stateless protocol: rest protocol will be tested by restarting server and checking if interactions survive.
     2. Used mainly with lightweight clients i.e. smartphone.
   * SOAP (Simple Object Access Protocol) - a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks.

Its purpose is to induce extensibility, neutrality and independence.

It uses XML Information Set for its message format, and d on application layer protocols, most often Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

* + 1. SOAP approach: has many rules to define message exchange, binding, operations between devices.
    2. Used majorly for testing heavyweight clients i.e. desktop api.
  + AJAX (Asynchronous Javascript And XML)

1. Jmeter is independent of OS of the machine, so Jmeter can run on any OS.
2. Other Performance Test Tools: LoadRunner (Micro Focus/HP), WebLoad (RadView), Rational Performance Tester (IBM), NeoLoad etc.
3. **Download** and install Java, then setup JDK path.
   * (User variable) JAVA\_HOME: C:\Program Files\Java\jdkx.x.x
   * (System variable) classpath: C:\Program Files\Java\jdkx.x.x\bin
4. If any issue in running jmeter tests:

Add variable “*JMETER\_HOME*” to the path “*D:\apache-jmeter-4.0*”

1. Always run test in NON-GUI Mode and export the report into csv file.
2. Jmeter needs heap of 1GB, depending on number of users this will vary.
3. **Download** Jmeter.zip file 🡪 unzip the file then,
4. GUI MODE
   * Open *bin/jmeter.bat* file i.e from bin folder ***Or***
   * Run *bin/ApacheJMeter.jar* i.e from bin folder
5. Non-GUI MODE
   * Open command prompt here “*C:\Users\Downloads\apache-jmeter-4.0\bin*” i.e path of jmeter bin folder.
   * Type command “*Java –jar ApacheJMeter.jar*”
6. A maximum of 2000 threads/users/request can be run with a 2GB RAM, 1GB java heap (approx.) which can be increased as per below link.

<https://www.wikihow.com/Increase-Java-Memory-in-Windows-7>

1. Install *plugins manager* to automate/ease installing plugins for jmeter.
2. <https://jmeter-plugins.org/wiki/PluginsManager/> download plugins manager JAR file
3. put it in the path “*lib/ext”*
4. Go to “*options*” from Jmeter GUI to access “plugins manager”
5. Best option to test webpages’ performance is to record the test script by
6. HTTP(s) recorder : use recording controller, setup proxy on browser, import root CA certificate
7. Blazemeter: log in to blazemeter, export test into jmx)/download and use Badboy software to record test.
8. **Jmeter Execution Order:**(order in which elements are executed in Test Plan)
   * Config Element – basic setup elements
   * Pre Processor – before running sampler
   * Timers
   * Sampler - Tests
   * Post Processor – after running sampler
   * Assertion – Verify expected with observed output
   * Listener – results viewer
9. Elements of Jmeter:

|  |  |
| --- | --- |
| **TERMS** | **USAGE** |
| Test plan | Container for whole test setup; |
| Thread Group | Number of Users |
| Sampler | Generates request |
| Listener | Shows test Results |
| Config Element | Default setup for respective requests. |
| Ramp up Period | Users added within time.  If ramp up=1, users =100 🡪100 users are added in 1 sec. |
| Loop count | Adding iteration of users. If loop=3, user=2  Each user will be added 3 times synchronously. |
| Latency | Time to first byte  (Time at which first byte of response is received which is required to render/display webpage). |
| Pre-Processor | Before executing sample request, preprocessor are executed.  Ex: initializing values before sampler runs the thread. |
| Post-Processor | After executing sample request, this is executed. |
| Timer | Used to delay execution of samplers.  The goal of timers is to simulate real user behaviour by waiting between the interactions with a web application. |
| Logic Controller | Used to break up the flow of execution and allow you to control the order that samplers are processed. |

1. Ways to optimize Server capacity:
   * Reduce TTFB (Time to first byte: is time taken for client/browser to access webpage from server and render the HTML file successfully).

|  |  |  |
| --- | --- | --- |
| TTFB | | |
| Browser sends HTTP request | Server processes the request | Server sends back the first byte of response to client. |

* + Increase bandwidth of Server (CPU, Memory, disk I/O etc.)
  + Enable Webpage Caching (use locally saved files to render page)
  + Improve HTML code: reduce large image sizes, use minified CSS, JS.
  + Reduce webpage redirects.
  + TLS: use security certificates effectively.

1. How to Debug Test:
   * Use “dummy sampler” for debugging/testing.
   * Enable Debug from Help.
   * To view debug logging, open jmeter.log under the bin directory or enable Log Viewer before you run it.
2. Google’s PageSpeed insights: <https://developers.google.com/speed/pagespeed/insights/> can be used to check TTFB and also options to enhance TTFB.

*Max response time <=200ms*, anything above this value makes page slow.

1. Variables, Functions, Properties are all *case sensitive.*
2. Variables: <https://jmeter.apache.org/usermanual/functions.html>

* User defined variable: (set once, use throughout)
* Test plan variable: (set once, use throughout)
* User parameters: (one per user)
* Variables in BeanShell script (external script)
* If variable is undefined, variable name will be returned there is no garbage value concept i.e., if $(var1) is undefined, the value returned when this variable is used = $(var1).

1. Functions:

* If function has parameters: ${function\_name(value1, value2)}
* If function has no parameters: ${function\_name}
* To escape comma in parameter name: ${function\_name(val\,1)

1. Properties: Properties are a way to move data from outside to test plan, from one thread group to other (thus properties are global)

* System property: properties/values inherited from java. (Values present in *system.properties* file).
* User Property: To overwrite any data in jmeter properties file, an user properties file can be created and used, more like treating jmeter as master data and user properties to overwrite master data instead of editing jmeter.properties file (Values present in *user.properties* file).
* Jmeter property: To pass data between threads/store data. (values present in *jmeter.properties* file)

1. White spaces are ignored in function/variable name.

${function (name1, name2)} ;

Second parameter = “name2” not “ name2”.

1. Jmeter uses Log4j framework to store logs.

* Logs will be stored under Log4j2.xml file, under “apache-jmeter-4.0\bin” or from “options/log viewer” from jmeter tool.
* Below are the logging levels:
  + None – Logging is turned off
  + Error – Runtime errors, severe errors, unexpected conditions.
  + Warn – Poor use of APIs, ‘almost errors’, unused variables etc.
  + Info – Run time events
  + Debug – Detailed flow information through system
* Use Log Level to check particular log levels(warn/error/… etc)

1. Distributed Testing: distribute tests over many slaves by Remote Testing thus increasing testing capability by making local machine as master and remote machines as slave.
   * Setup Slave in Master machine–
     1. Open apache-jmeter-4.0/bin/jmeter.properties
     2. Add remote\_hosts i.e., assign slaves here.

Ex: remote\_hosts=xx.xx.xx.xx, yy.yy.yy.y

* + Run “create-rmi-keystore.bat” file (go to bin folder of jmeter and run)

1. Enter username: rmi
2. Enter other details whatever you want.
3. Enter password: changit
4. A filewill be generated in bin directory “rmi\_keystore.jks”
5. Now restart the jmeter to apply changes.
   * Run “jmeter-server.bat” file (windows) on slave system, then run the test.
   * Run Slaves from Master –
     + 1. Go to local machine/master
       2. To run test in GUI-MODE, go to Run 🡪 Remote Start 🡪 added Slaves i.e., IP addresses will be here, run it.
       3. To run test in NON-GUI MODE, type in command

$ jmeter –n –t “[path of jmx file]” –l “[path of csv file]” –R [slave\_ip1]

OR

go to bin folder of jmeter and type below command,

$ java –jar ApacheJMeter.jar –n –t “[path of jmx file]” –l “[path of csv file]” –R [slave\_ip1]

1. Server Monitor Health Check with PerfMon plugin:
   * Checking Server (machine hosting app) performance during testing.
   * Install perfmon plugin from plugin manager/download perfmon add it to jmeter lib and ext folder of client/local machine.
   * Download and unzip “perfMon server agent” (ServerAgent-2.2.3.zip) on server where WebApp is hosted. <https://github.com/undera/perfmon-agent/blob/master/README.md>
   * Run “startAgent.bat” file on server, connection on port 4444 will be started.
   * From client/local machine check if the server is accepting connection at port 4444 by below command.

$ Telnet IP\_Address 4444 🡪 on server accepting connection notification can be seen.

* + Create jmeter test to monitor server health on client.

1. Steps to enhance/Increase Threads/users:
   * Increase java heap size of machine running jmeter

<https://www.wikihow.com/Increase-Java-Memory-in-Windows-7>

* + Don’t use any Listeners
  + Increase RAM size of machine
  + $ Jmeter --? Or Jmeter –h (after entering bin folder)

🡪 Displays all available commands for Jmeter.

* + Use Non-GUI mode to run Tests

$ jmeter –n –t [jmeter file] –l [results file] –e –o [path to results file]

{Ex: jmeter –n –t http.jmx –l http.csv –e –o /desktop/folder1}

OR

$ Java –jar ApacheJMeter.jar –n –t http.jmx –l /desktop/folder1/http.csv;

Jmeter 🡪 use jmeter.bat in windows

-n 🡪 non gui mode

-t 🡪 path of jmeter file containing test plan

-l 🡪 JTL file/csv file to log results

-j 🡪 log filename

-e 🡪 generate report after load test

-o 🡪 path to output file

NOTE: under output file folder open “index.html” to view HTML report of o/p.

## Controller

<http://jmeter.apache.org/usermanual/component_reference.html>

Used to customize order of execution of samplers/child elements.

* + Simple C: basic controller, used to group requests/controllers that’s all.
  + Loop C: multiples request i.e. loop count of Thread multiplied with loop count of loop controller.
  + Recording C: used only with HTTPs TestScript recorder to store all the recordings.
  + Interleave C: Interleave Controller allow us to pick and execute a single child element out of multiple child orderly in each loop iteration.

Ex: if there are 3 requests inside interleave & thread loopcount=3, in first loop request1 will be executed, then request2 in second loop, then req3 in third loop etc.

* + Random Order C: child elements of this Controller will be executed in random order.
  + Random C: similar to random order controller, picks a single child element in each iteration but unlike Interleave controller it picks the child element randomly.
  + Critical Section C: child elements of this controller are accessed by only one thread at a time.
  + Runtime C:  used to limit the time of execution of its child elements.

Example: If runtime=100 sec; elements inside the Runtime controller will run for 100 seconds with as much iteration as possible.

* + Once Only C: used in situation where we would like to perform an operation only once per thread even if the operation is executed in a loop. Thus, request is executed during first iteration of a thread.

Example: during login to a webpage.

* + ForEach C: used to perform request in loops, based on the values of a set of related variables.
  + If C: The child element inside it will run or not based on a condition which should evaluate to true or false.
  + While C: used to run the child elements inside it till the value specified in its control panel is evaluated to false.
  + Include C: used to provide modularity, i.e. we can add an external .jmx file to our existing script.
  + Switch C: used to pick one element for processing out of its multiple child elements. The elements are not picked in sequential/random order, instead it is based on switch value defined in its control panel.
  + Module C: used to reuse a test fragment (e.g. a sampler) into our script again by selecting the module.
  + Throughput C: allows the user to control requests under this, on how often requests are executed.
    1. **Percent executions:** execute a certain percentage of iterations.

**Ex1:** If percentage=50, request inside throughput=2, then only 50% of each request will be executed thus abrupt execution.

* + 1. **Total executions:** stop executing after a certain number of executions have occurred.

**Ex:** if total execution throughput =2, each request inside throughput controller will be executed 2 times irrespective of loop count of thread.

* + Transaction C: used to group multiple sampler requests into one.

## Sampler

Jmeter has Ability to load and test performance of many different applications/server /protocol types:

1. Web - [HTTP](#_HTTP:), HTTPS (Java, NodeJS, PHP, ASP.NET, …)
2. [SOAP / REST API Web Services.](#_SOAP/REST_API:)
3. [FTP](#_FTP:)
4. [DB via JDBC](#_DB:)
5. [JUnit](#_JUNIT:) – to use Junit test classes
6. LDAP (Lightweight Directory Access Protocol)
7. MOM (Message-oriented middleware) via JMS (java messaging service): One to one and one to many messaging services are tested with Apache ActiveMQ opensource message broker. <https://www.tutorialspoint.com/jmeter/jmeter_jms_test_plan.htm>
8. Mail - SMTP(S), POP3(S) and IMAP(S)
9. Native commands or shell scripts
10. TCP Sampler
11. JSR223 – to use scripting language
12. Bean Shell – to use Apache Bean scripting language
13. Java request sampler – to use our own developed java classes
14. Debug Sampler –
15. Access Log –

### HTTP:

### FTP:

### SOAP/REST API:

### DB:

### JUNIT:

## **Pre-Processor**

## **Post Processor**

## **Assertion**

## **Timer**

## **Listener**

## IMPORTANCE OF PERFORMANCE TESTING:

* + Example when Ecommerce sites enables a festive season with discounts, people are expected to login and check products and order.

Thus a huge number of users will be signing up, logging is, adding product to cart, and browsing products, paying for order etc. at this point we need to make sure the webpage can handle these many users with different combination can be handled by server.

Thus to create these users Virtually performance testing tool Is used just like selenium tool to simulate artificial users but mainly to measure performance of servers and not the behaviour/functions of the webpage.