Performance Testing With JMeter

Tejas Parikh (t.parikh@northeastern.edu)

CSYE 6225, Fall 2019
Northeastern University

Why Performance Testing

- Right size infrastructure
  - Handle peak load
  - Availability
  - Response Times
  - Throughput
  - Utilization
- Identify bottlenecks
  - Application
  - Database
  - Integrations with external services
- Capacity planning

What is JMeter?

- JMeter is an software that can be used to execute performance testing, load testing and functional testing of your web applications.
- JMeter can also simulate a heavy load on a server by creating tons of virtual concurrent users to web server.
- JMeter is
  - Open Source
  - Easy to use
  - Platform independent
  - Flexible
  - Supports multiple protocols

Installing JMeter

- https://jmeter.apache.org/usermanual/get-started.html#lets_start

JMeter Test Plans

- A test plan describes a series of steps JMeter will execute when run.
- A complete test plan will consist of one or more Thread Groups, logic controllers, sample generating controllers, listeners, timers, assertions, and configuration elements.

Elements of a Test Plan

- Thread Group
- Controllers
- Samplers
- Listeners
- Timers
- Assertions
- Configuration Elements
- Pre-Processor Elements
- Post-Processor Elements

Thread Group

- Thread group elements are the beginning points of any test plan.
- All controllers and samplers must be under a thread group.
- Other elements, e.g. Listeners, may be placed directly under the test plan, in which case they will apply to all the thread groups.
- As the name implies, the thread group element controls the number of threads JMeter will use to execute your test.
- The controls for a thread group allow you to:
  - Set the number of threads
  - Set the ramp-up period
  - Set the number of times to execute the test
- Each thread will execute the test plan in its entirety and completely independently of other test threads.
- Multiple threads are used to simulate concurrent connections to your server application.
- By default, the thread group is configured to loop once through its elements.

Controllers

- JMeter has two types of Controllers: Samplers and Logical Controllers. These drive the processing of a test.
- *Samplers* tell JMeter to send requests to a server. For example, add an HTTP Request Sampler if you want JMeter to send an HTTP request.
- *Logical Controllers* let you customize the logic that JMeter uses to decide when to send requests. For example, you can add an Interleave Logic Controller to alternate between two HTTP Request Samplers.

Samplers

- Samplers tell JMeter to send requests to a server and wait for a response. They are processed in the order they appear in the tree. Controllers can be used to modify the number of repetitions of a sampler. JMeter samplers include:
  - FTP Request
  - HTTP Request (can be used for SOAP or REST Webservice also)
  - JDBC Request
  - Java object request
  - JMS request
  - JUnit Test request
  - LDAP Request
  - Mail request
  - OS Process request
  - TCP request

Listeners

- Listeners provide access to the information JMeter gathers about the test cases while JMeter runs.
- The Graph Results listener plots the response times on a graph.
- The "View Results Tree" Listener shows details of sampler requests and responses, and can display basic HTML and XML representations of the response. Other listeners provide summary or aggregation information.
- Additionally, listeners can direct the data to a file for later use.
- Every listener in JMeter provides a field to indicate the file to store data to.
- Note that all Listeners save the same data; the only difference is in the way the data is presented on the screen.
- Listeners can be added anywhere in the test, including directly under the test plan. They will collect data only from elements at or below their level.

Timers

- By default, a JMeter thread executes samplers in sequence without pausing. It is recommend that you specify a delay by adding one of the available timers to your Thread Group. If you do not add a delay, JMeter could overwhelm your server by making too many requests in a very short amount of time.
- A timer will cause JMeter to delay a certain amount of time before each sampler which is in its scope.
- If you choose to add more than one timer to a Thread Group, JMeter takes the sum of the timers and pauses for that amount of time before executing the samplers to which the timers apply.
- Timers can be added as children of samplers or controllers in order to restrict the samplers to which they are applied.
- To provide a pause at a single place in a test plan, one can use the Flow Control Action Sampler.

Assertions

- Assertions allow you to assert facts about responses received from the server being tested.
- Using an assertion, you can essentially "test" that your application is returning the results you expect it to.

Configuration Elements

- A configuration element works closely with a Sampler. Although it does not send requests (except for HTTP(S) Test Script Recorder), it can add to or modify requests.
- A configuration element is accessible from only inside the tree branch where you place the element.

Pre-Processor Elements

- A Pre-Processor executes some action prior to a Sampler Request being made.
- If a Pre-Processor is attached to a Sampler element, then it will execute just prior to that sampler element running.
- A Pre-Processor is most often used to modify the settings of a Sample Request just before it runs, or to update variables that aren't extracted from response text.

Post-Processor Elements

- A Post-Processor executes some action after a Sampler Request has been made.
- If a Post-Processor is attached to a Sampler element, then it will execute just after that sampler element runs.
- A Post-Processor is most often used to process the response data, often to extract values from it.

Element Execution Order

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

Using Variables to Parameterise Tests

Variables don't have to vary - they can be defined once, and if left alone, will not change value. So you can use them as short-hand for expressions that appear frequently in a test plan. Or for items which are constant during a run, but which may vary between runs. For example, the name of a host, or the number of threads in a thread group.

When deciding how to structure a Test Plan, make a note of which items are constant for the run, but which may change between runs. Decide on some variable names for these - perhaps use a naming convention such as prefixing them with **C_** or **K_** or using uppercase only to distinguish them from variables that need to change during the test. Also consider which items need to be local to a thread - for example counters or values extracted with the Regular Expression Post-Processor. You may wish to use a different naming convention for these.

For example, you might define the following on the Test Plan:

```
HOST                www.example.com
THREADS             10
LOOPS               20
```

You can refer to these in the test plan as **${HOST} ${THREADS}** etc. If you later want to change the host, just change the value of the **HOST** variable. This works fine for small numbers of tests, but becomes tedious when testing lots of different combinations. One solution is to use a property to define the value of the variables, for example:

```
HOST                ${__P(host,www.example.com)}
THREADS             ${__P(threads,10)}
LOOPS               ${__P(loops,20)}
```

You can then change some or all of the values on the command-line as follows:

```
jmeter … -Jhost=www3.example.org -Jloops=13
```

JMeter User Manual

- https://jmeter.apache.org/usermanual/index.html

Additional Resources

https://fall2019.csye6225.cloud/