JMeter

# Difference between Performance testing, Load testing, and Stress Testing:

<https://www.guru99.com/performance-vs-load-vs-stress-testing.html>

<https://www.blazemeter.com/blog/performance-testing-vs-load-testing-vs-stress-testing/>

<https://www.softwaretestinghelp.com/what-is-performance-testing-load-testing-stress-testing/>

Performance testing is the superset for both load & stress testing. My personal usage of these 3 types of tests are:

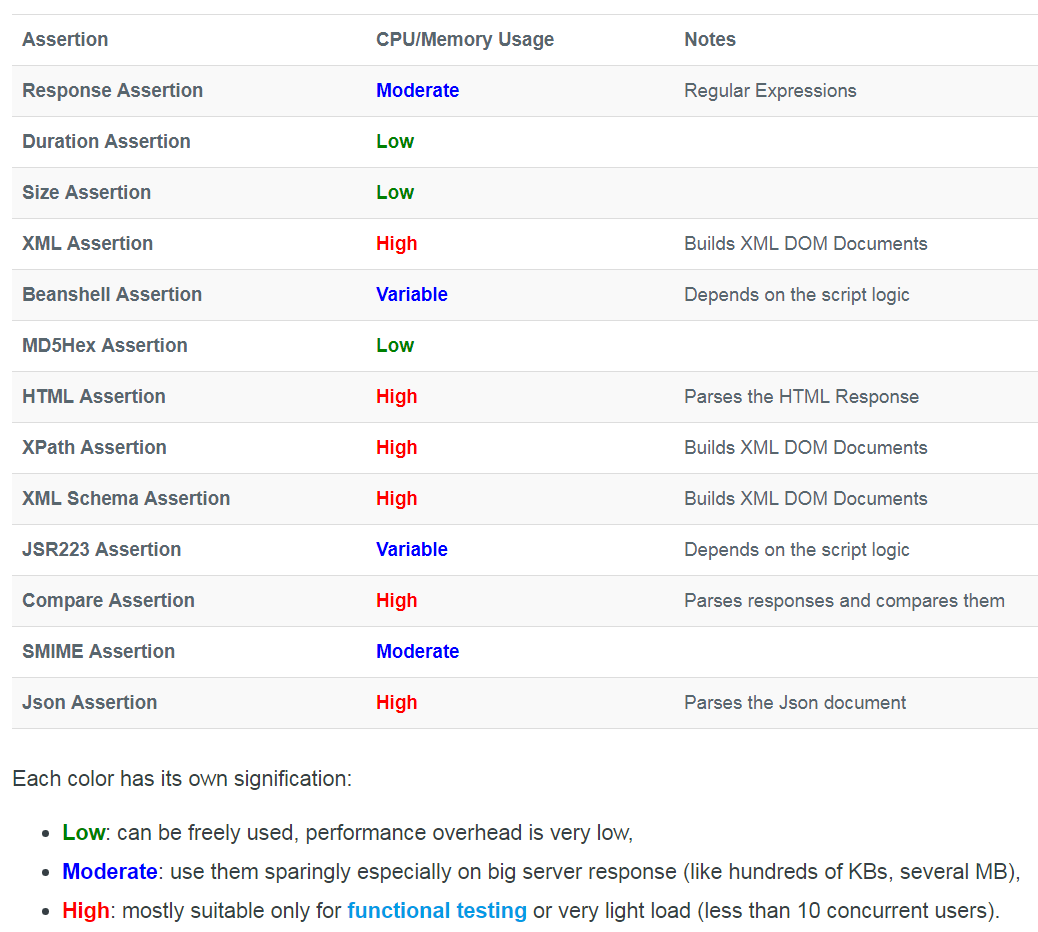
* Performance test: focus on speed or latency, such as response time, and script execution duration.
* Load test: focus on concurrent users. Normally you ramp up to the targeted number of users in a certain period (constantly and steadily increasing the load, not a spike), then keep the same number of users for another period, then release the users gradually.
* Stress test: including spike test and soak test, but both goals is negative tests, i.e., to make the system fail, and observe how the system will recover after failure. On the other hand, load test is to determine the upper limit.

# Difference usage between Postman and JMeter:

I get both functional and performance test done with postman. Compare to JMeter, postman is more user intuitive and easier for me to ramp up manual testers to start to write scripts in postman to test API functions. During development, functional test always comes before performance tests and load/stress tests. Generally, the load and stress tests get developed after most of the functionalities are settled.

To do performance test with postman is easy, you can choose to use its embedded snippet to add assertion for each http calls response time, which I find it will have low ROI, due to high time consumed with low returns. My decision is only to assert how long each script runs. When performance get degraded, could just because a left join in the DB, a 15 second script can take 1 minute to finish. In the pre-request script of the 1st http call of the script, I capture the time, and in the test script of the last http call of the script, I capture the time again. The difference should below 15 seconds \* 2 (I give 100% variance, since when there is a performance degradation, the duration normally at least doubled). Also, I don’t add this kind of check for all the scripts, only for longer and complicated scripts.

The reason that JMeter is not suitable for functional test is also because that functional test will have a lot of JSON assertions, which according to the chart below, will be high memory and CPU consuming. On the other hand, the duration assertion in JMeter has low CPU/Memory usage, which means JMeter is designed for load tests but not functional test.



# How to use JMeter:

To use JMeter for load testing, in order to maximize the concurrent users before your computer crashes, you have to run JMeter in cmd, not in GUI mode, and do other modifications as mentioned in: <https://www.blazemeter.com/blog/what%E2%80%99s-the-max-number-of-users-you-can-test-on-jmeter/>. If you really need to simulate tons of users which are over the limit of what JMeter can support, then use Blazemeter, which is JMeter on cloud: you can get as many concurrent uses as you want, and from anywhere of the world that you want, but it comes with a cost.

There are 5 things that important in JMeter:

* Thread Group: define how many concurrent users.
* Samplers: HTTP request, FTP request, JDBC request, etc.
* Listeners: display results in Graphs, Table, Tree, and Log. If you want html version of reports, you can use the JMeter dashboard extensions: <https://jmeter.apache.org/usermanual/generating-dashboard.html>
* Configurations: csv Data driven tests (parameters), HTTP cookie manager, HTTP request defaults (such as URL), Login config element (log in username and password).
* Assertions: different types of assertions as displayed in the picture above.

Please refer to page 49 to 66 of “APItest.ppsx” for detailed instructions of how to use the above 5 elements in JMeter, it also includeds how to run JMeter in cmd (to be able to run script in cmd is the fundamental requirement to integrate QA tests in CICD pipeline). The “APItest.ppsx” was created by a former manual tester after I ramp her up for API testing, its content also included Graphic DB, Git code commit, and Postman, etc.

Note you can also use [on-line free tool](https://loadium.com/postman-to-jmeter-converter/) to convert your Postman script to JMeter script but you will lose all your variables and assertions, I find it will be easier to just create scripts in JMeter compare to convert from Postman. Since I use Postman and JMeter with different goals, so the scripts in JMeter does not need to be as complicated as the scripts I created in Postman.

# Simulation of real load:

If you want to do a good job for load and stress testing, the key to remember is, you suppose to simulate real use cases, not some use cases you imagined! Ask your marketing and operation friends what kind the challenges they faced or will be facing. The load testing for a huge ecommerce website which many face big sales in black Fridays, will be different than a IoT company has all photo sensors installed in the fields that will report light changes during each sun rise.