27th Oct

Performance Testing:

Performance Testing is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload.

The goal of Performance Testing is not to find bugs but to eliminate performance bottlenecks(Upto to level of bottle means zoom meeting is having limit 500 users if 501 users will join then the 501 is our bottlenecks). The focus of Performance Testing is checking a software program's

The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. It is a subset of performance engineering and is also known as ***"Perf Testing"***.

- **Speed** – Determines whether the application responds quickly
- **Scalability** – Determines the maximum user load the software application can handle.
- **Stability** – Determines if the application is stable under varying loads

Types of Performance Testing

- **Load testing –** checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
- **Stress testing –** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
- **Endurance testing –** is done to make sure the software can handle the expected load over a long period of time.
- **Spike testing –** tests the software's reaction to sudden large spikes in the load generated by users.
- **Volume testing** – Under Volume Testing large no. of. Data is populated in a database, and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes.
- **Scalability testing** – The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

Steps of Performance Testing

Step 1) Identify Your Testing Environment

Know your physical test environment, production environment and what testing tools are available. Understand details of the hardware, software and network configurations used during testing before you begin the testing process. It will help testers create more efficient tests. It will also help identify possible challenges that testers may encounter during the performance testing procedures.

Step 2) Identify the Performance Acceptance Criteria

This includes goals and constraints for throughput, response times and resource allocation. It is also necessary to identify project success criteria outside of these goals and constraints. Testers should be empowered to set performance criteria and goals because often the project specifications will not include a wide enough variety of performance benchmarks. Sometimes there may be none at all. When possible finding a similar application to compare to is a good way to set performance goals.

Step 3) Plan & Design Performance Tests

Determine how usage is likely to vary amongst end users and identify key scenarios to test for all possible use cases. It is necessary to simulate a variety of end users, plan performance test data and outline what metrics will be gathered.

Step 4) Configuring the Test Environment

Prepare the testing environment before execution. Also, arrange tools and other resources.

Step 5) Implement Test Design

Create the performance tests according to your test design.

Step 6) Run the Tests

Execute and monitor the tests.

Step 7) Analyze, Tune and Retest

Consolidate, analyze and share test results. Then fine tune and test again to see if there is an improvement or decrease in performance. Since improvements generally grow smaller with each retest, stop when bottlenecking is caused by the CPU. Then you may have the consider option of increasing CPU power.

Tools Of Performance testing

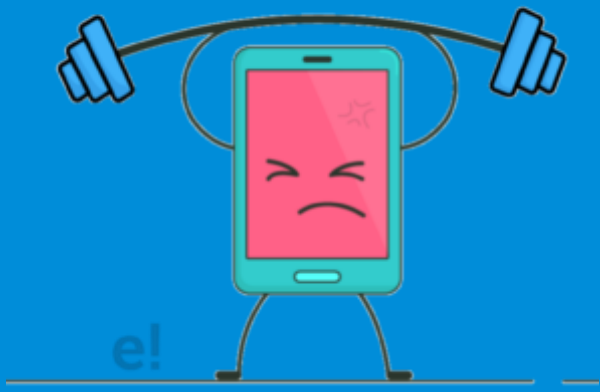
1. LoadNinja
2. Apache JMeter
3. WebLOAD
4. LoadUI Pro
5. LoadView
6. NeoLoad
7. LoadRunner
8. Silk Performer
9. AppLoader
10. SmartMeter.io

Difference between Jmeter and LoadRunner

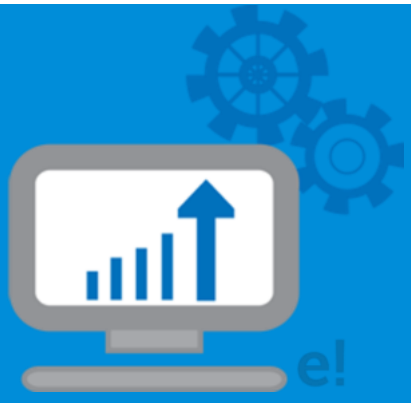
	Availability
	

Apache JMeter is an open source, free software testing tool that can be easily downloaded for performing any test.	LoadRunner is an expensive software testing tool. It has recently released free trial versions but it cannot be simply downloaded for use.

Load Generation Capacity

	
JMeter has an unlimited load generation capacity.	LoadRunner has a limited load generation capacity.

Execution

	
Execution is easier in JMeter. You just need to install Java, download JMeter and upload the JMeter script file.	Execution is complex as compared to JMeter. It creates one thread for each user.

Analysis Report

Results are easy to understand for less experienced engineers, and also allow in-depth analysis for testers with more know-how.

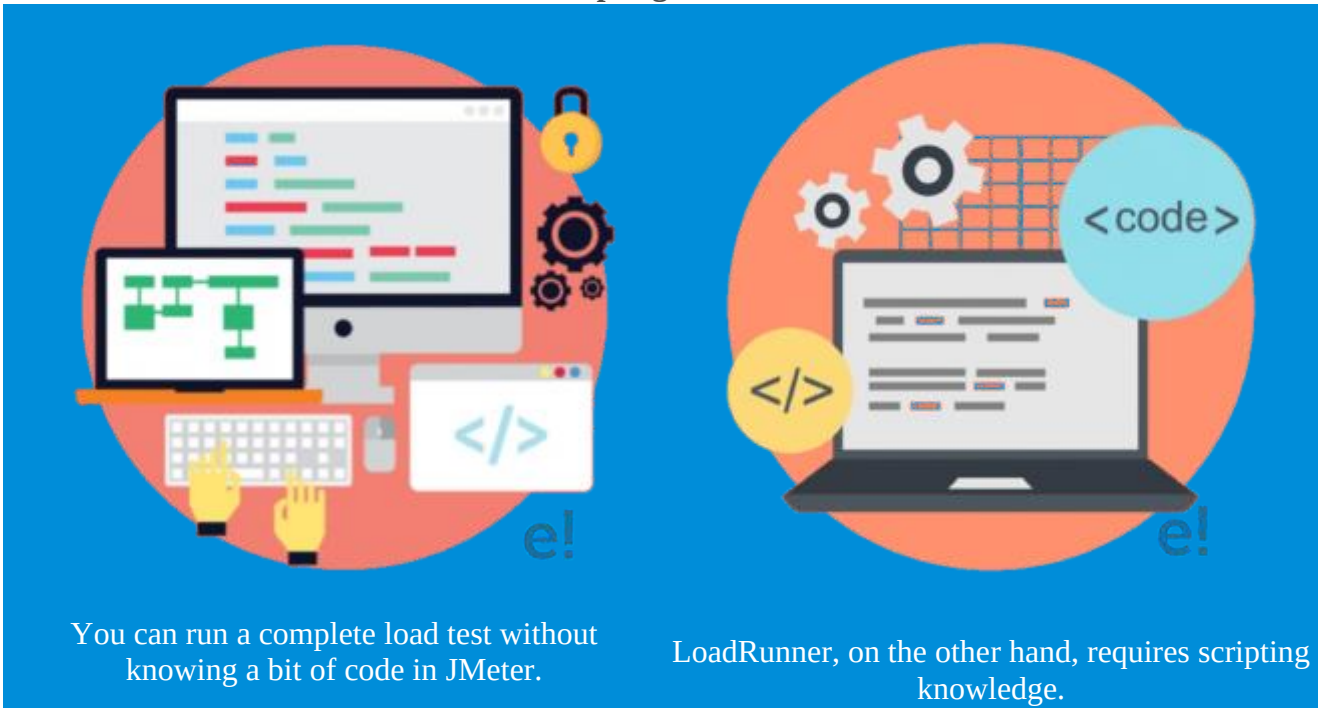
The information is in a raw format which is parsed by HP Analysis to generate various graphs.

Open-Source & Community Support

It has vibrant community and supports users who run into issues and problems.

It is owned by a large corporate, limiting the number of users.

Scripting



You can run a complete load test without knowing a bit of code in JMeter.

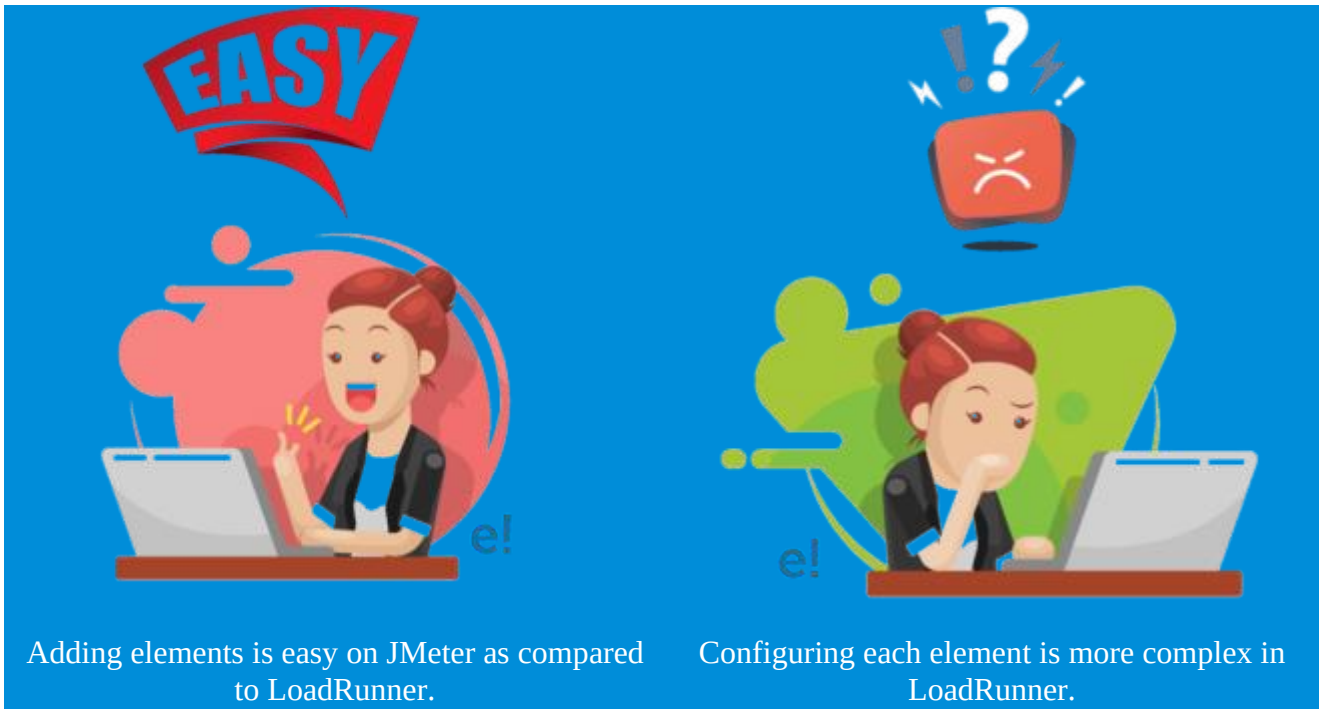
LoadRunner, on the other hand, requires scripting knowledge.

Building Test Scenarios

JMeter doesn't require adding beginning or ending transaction elements.

LoadRunner is more complex because it requires managing different agents.

Elements



Adding elements is easy on JMeter as compared to LoadRunner.

Configuring each element is more complex in LoadRunner.