## What is performance testing?

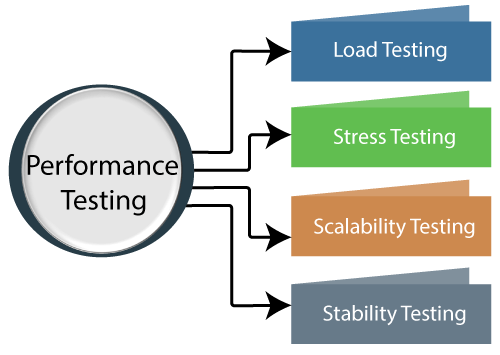
Checking the behavior of an application by applying some load is known as performance testing.

Performance testing is in general a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload

Types of Performance Testing

Following are the types of performance testing:

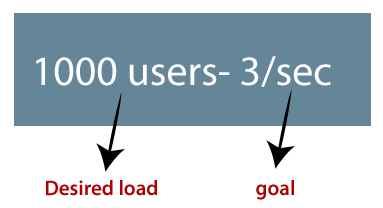
* **Load testing**
* **Stress testing**
* **Scalability testing**
* **Stability testing**



### **1.Load testing**

check the performance of an application by applying some load which is either less than or equal to the desired load is known as load testing.

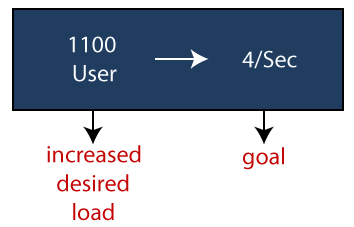
**For example:** In the below image, **1000 users** are the **desired load**, which is given by the customer, and **3/second** is the **goal** which we want to achieve while performing a load testing.



### **2.Stress Testing**

The stress testing is testing, which checks the behavior of an application by applying load greater than the desired load.

**For example:** If we took the above example and increased the desired load 1000 to 1100 users, and the goal is 4/second. While performing the stress testing in this scenario, it will pass because the load is greater (100 up) than the actual desired load.



### **3.Scalability Testing**

Checking the performance of an application by increasing or decreasing the load in particular scales (no of a user) is known as **scalability testing**. Upward scalability and downward scalability testing are called scalability testing.

Scalability testing is divided into two parts which are as follows:

* **Upward scalability testing**
* **Downward scalability testing**

**a)Upward scalability testing**

It is testing where we **increase the number of users on a particular scale** until we get a crash point. We will use upward scalability testing to find the maximum capacity of an application.

**b)Downward scalability testing**

The downward scalability testing is used when the load testing is not passed, then start **decreasing the no. of users in a particular interval** until the goal is achieved. So that it is easy to identify the bottleneck (bug).

### **4.Stability Testing**

Checking the performance of an application by **applying the load for a particular duration of time** is known as **Stability Testing**.

## Performance testing example

Let us take one example where we will **test the behavior of an application where the desired load is either less than 1000 or equal to 1000 users**.

In the below image, we can see that the **100 up** users are increased continuously to check the **maximum load**, which is also called **upward scalability testing**.

* **Scenario1:** When we have the 1000 users as desired load, and the 2.7/sec is goal time, these scenarios will pass while performing the load test because in load testing, we will concentrate on the no. of users, and as per the requirement it is equal to 1000 user.
* **Scenario2:** In the next scenario, we will increase the desired load by 100 users, and goal time will go up to 3.5\sec. This scenario will pass if we perform stress testing because here, the actual load is greater than (1100) the desired load (1000).
* **Scenario3:** In this, if we increase the desired load three times as  
  **1200 → 3.5\sec:** [it is not less than or equal to the desired load that's why it will **Fail**]  
  **1300 → 4\sec:** [it is not less than or equal to the desired load. i.e., **Fail**]  
  **1400 → Crashed**

#### **Note1: Volum**Performance Testing

**Fig:-Performance Testing Example**

**The Seven Performance Testing Steps**

* **Identify The Testing Environment:-** First, check your hardware, software, network, and tools requirements and make sure they are present.
* See if there is anything else the testing team needs to design and perform the test. Also, try to look for the first performance test challenges.
* As a result, all these initial steps will help in creating more efficient tests. They will make things easier for the tester to counter the issues they might face during the testing.
* **Identity Performance Metrics:-** You need to set the goals for your test and constraints for throughput and the machine’s minimum response time and the time it should take for the resource allocation. You need to identify the success areas of your project outside the constraints that you have set up.
* **Design & Plan Performance Tests:-** Once the goals are set up, you have identified all the necessary factors to make a suitable testing environment for your system. You need to plan out the test scenarios based on the goals you have chosen. It would be best to determine how amongst the different users and case scenarios, the user will be divided. It is required that you use various simulation and test cases when performing stress tests on the system. Plan out the performance data and outline which metrics need to be reported.
* **Configure The Testing Environment:-** Now your setup is almost complete. What remains is configuring the testing environment to execute your tests and arranging the necessary tools along with the essential resources.
* **Begin Implementing Your Test Designs:-**Here you need to create the performance test in your system according to the plan. And the design which you have made and prepare your system for the tests.
* **Execute The Tests:-** Begin the execution of the test and start monitoring the performance dips of your system.
* **Analyze, Tune, And Then Retest:-** Once the tests are over, consolidate the test reports and share the result with the team. After that, start fine-tuning your system and rerun the tests. To check if there is any improvement in the performance after the tuning is done.