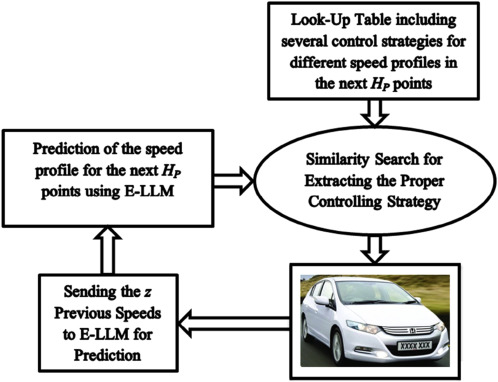
**Utilization of Testing tools**

****

**DEFINITION**

**performance testing**

****

By

* [**Alexander S. Gillis,**](https://www.techtarget.com/contributor/Alexander-S-Gillis)Technical Writer and Editor

Performance testing is a testing measure that evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload. Organizations will run performance tests in order to identify performance-related [bottlenecks](https://www.techtarget.com/searchnetworking/definition/bottleneck).

Without some form of performance testing in place, system performance will likely be affected with slow response times, experiences that are inconsistent between users and the operating system, creating an overall poor user experience. Determining if the developed system meets speed, responsiveness and stability requirements while under workloads will help ensure a more positive user experience.

Performance testing can involve quantitative tests done in a lab, or in some scenarios, occur in the production environment. Performance [requirements should be identified](https://www.techtarget.com/searchsoftwarequality/answer/Required-prerequisites-for-performance-testing) and tested. Typical parameters include processing speed, data transfer rates, network [bandwidth](https://www.techtarget.com/searchnetworking/definition/bandwidth) and [throughput](https://www.techtarget.com/searchnetworking/definition/throughput), workload efficiency and reliability. As an example, an organization can measure the response time of a program when a user requests an action; the same can be done at scale. If the response times are so slow that it would annoy end users, then this means it should be tested to find where the bottleneck is.

**Why use performance testing?**

An organization can use performance testing as a diagnostic aid to locate computing or communications bottlenecks within a system. Bottlenecks are a single point or component within a system's overall function that holds back overall performance. For example, even the fastest computer will function poorly on the web if the bandwidth is less than 1 megabit per second ([Mbps](https://www.techtarget.com/searchnetworking/definition/Mbps)). Slow data transfer rates might be inherent in hardware but could also result from software-related problems -- such as too many applications running at the same time or a corrupted file in a web browser.

Performance testing can be used as a form of software testing to help identify the nature or location of a software-related performance problem by highlighting where an application might fail or lag. This form of testing also can be used to ensure an organization is prepared for a predictable major event, for online stores.

Performance testing can also verify that a system meets the specifications claimed by its manufacturer or vendor. The process can be used to compare two or more devices or programs.

**Performance testing metrics**

A number of performance metrics, or key performance indicators ([KPIs](https://www.techtarget.com/searchbusinessanalytics/definition/key-performance-indicators-KPIs)), can help an organization evaluate current performance.

Performance metrics commonly include:

* **Throughput.** How many units of information a system processes over a specified time
* [**Memory**](https://www.techtarget.com/searchstorage/definition/memory-card)**.** The working storage space available to a processor or workload
* **response time, or latency.** The amount of time that elapses between a user-entered request and the start of a system's response to that request
* **Bandwidth.** The volume of data per second that can move between workloads, usually across a network
* **CPU interrupts per second.** The number of hardware interrupts a process receives per second

These metrics and others help an organization perform multiple types of performance tests.

**How to conduct performance testing**

Because performance testing can be conducted with different types of metrics, the actual process can vary greatly. However, a generic process may look like:

1. **Identifying the testing environment.** This includes test and production environments as well as the testing tools.
2. **Identifying and defining acceptable performance criteria.** This should include performance goals and constraints for metrics.
3. **Planning the performance test.** Test all possible use cases. Build test cases around performance metrics.
4. **Configuring and implementing test design environment.** Arrange resources to prepare the test environment, then begin to implement it.
5. **Running the test.** The test also should be monitored.
6. **Analyzing and retesting.** Look over the results. After any fine-tuning, retest to see if there is an increase or decrease in performance.

Organizations should find testing tools that can best automate their performance testing process. In addition, changes should not be made to the testing environments between tests.

**Types of performance testing**

There are two main performance testing methods: [load testing](https://www.techtarget.com/searchsoftwarequality/definition/load-testing) and [stress testing](https://www.techtarget.com/searchsoftwarequality/definition/stress-testing). However, there are other types of testing methods that can be used to determine performance. Some examples are as follows:

* **Load testing** helps developers understand the behavior of a system under a specific load value. In the load testing process, an organization simulates the expected number of concurrent users and transactions over a duration of time to verify expected response times and locate bottlenecks. This type of test helps developers determine how many users an application or system can handle before that app or system goes live. Additionally, a developer can load test specific functionalities of an application, such as a checkout cart on a webpage. A team can include load testing as part of a [continuous integration](https://www.techtarget.com/searchsoftwarequality/definition/continuous-integration) (CI) process, in which they immediately test changes to a code base through the use of automation tools, such as [Jenkins](https://www.techtarget.com/searchsoftwarequality/definition/Jenkins).
* **Stress testing** places a system under higher-than-expected traffic loads so developers can see how well the system works above its expected capacity limits. Stress tests have two subcategories: soak testing and spike testing. Stress tests enable software teams to understand a workload's scalability. Stress tests put a strain on hardware resources in order to determine the potential breaking point of an application based on resource usage. Resources could include CPUs, memory and hard disks, as well as solid-state drives. System strain can also lead to slow data exchanges, memory shortages, data corruption and security issues. Stress tests can also show how long KPIs take to return to normal operational levels after an event. Stress tests can occur before or after a system goes live. A kind of production-environment stress test is called [chaos engineering](https://www.techtarget.com/searchitoperations/definition/chaos-engineering) and there are specialized tools for it. An organization might also perform a stress test before a predictable major event, such as Black Friday on an e-commerce application, approximating the expected load using the same tools as load tests.
* **Soak testing**, also called endurance testing, simulates a steady increase of end users over time to test systems' long-term sustainability. During the test, the test engineer monitors KPIs, such as memory usage, and checks for failures, such as memory shortages. Soak tests also analyze throughput and response times after sustained use to show if these metrics are consistent with their status at the beginning of a test.
* **Spike testing**, another subset of stress testing, assesses the performance of a system under a sudden and significant increase of simulated end users. Spike tests help determine if a system can handle an abrupt, drastic workload increase over a short period of time, repeatedly. Similar to stress tests, an IT team typically performs spike tests before a large event in which a system will likely undergo higher than normal traffic volumes.
* **Scalability testing** measures performance based on the software's ability to scale up or down performance measure attributes. For example, a scalability test could be performed based on the number of user requests.
* **Capacity testing** is similar to stress testing in that it tests traffic loads based on the number of users but differs in the amount. Capacity testing looks at whether a software application or environment can handle the amount of traffic it was specifically designed to handle.