PERFORMANCE TESTING, LOAD TESTING & STRESS TESTING EXPLAINED

Performance testing is key for understanding how your system works. Without good performance testing, you don't know how your system will deal with expected—or unexpected—demands.

Load testing and stress testing are two kinds of performance testing. Knowing when to use load testing and when to use stressing testing depends on what you need:

- Do you need to understand how your system performs when everything is working normally?
- Do you want to find out what will happen under unexpected pressure like traffic spikes?

To ensure that your systems remain accessible under peak demand, run your system through performance testing.

Let's take a look.

Performance vs load vs stress testing

Performance testing is an umbrella term for load testing and stress testing.

When developing an application, software, or website, you likely set a benchmark (standard) for performance. This covers what happens under:

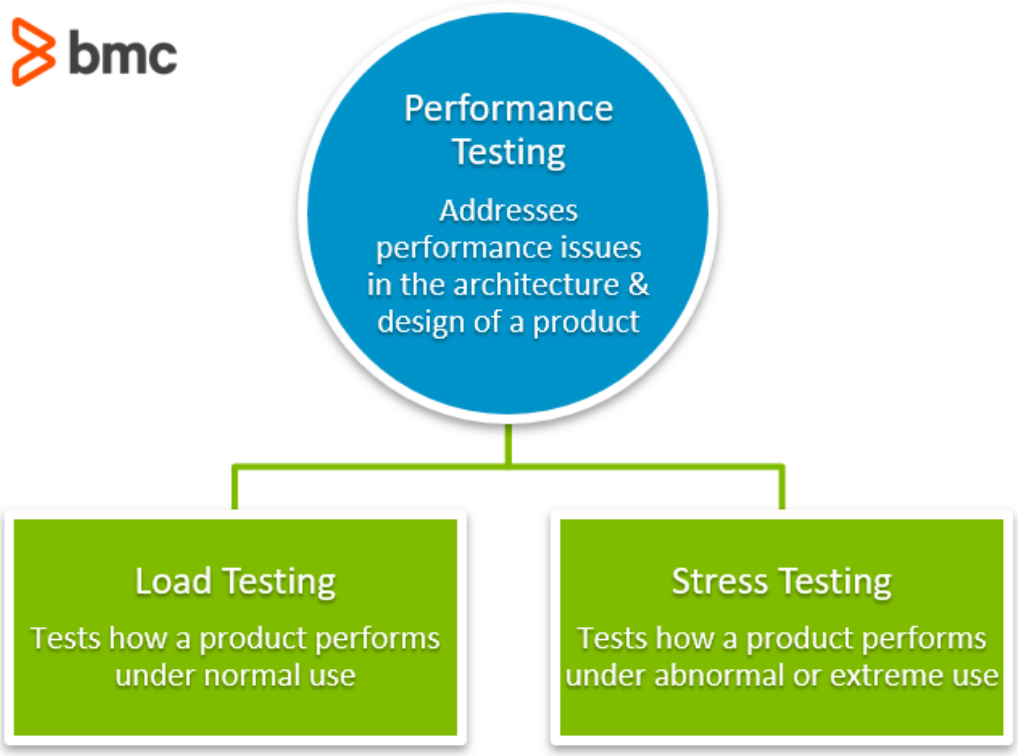
- **Regular parameters:** If everything goes as planned, how does it work?

- **Irregular parameters:** Can my website application survive a DDoS attack?

Load testing is testing how an application, software, or website performs when in use under an expected load. We intentionally increase the load, searching for a threshold for good performance. This tests how a system functions when it faces normal traffic.

Stress testing is testing how an application, software, or website performs when under extreme pressure—an unexpected load. We increase the load to its upper limit to find out how it recovers from possible failure. This tests how a system functions when it faces abnormal traffic.

Now let's look at each in more detail.



What is performance testing?

When you want to find out how the components of a system are performing under a given situation, performance testing is the way to go. You'll want to find out three key factors about your product:

- The scalability
- The resource usage
- The reliability

As a subset of performance engineering, this type of testing addresses performance issues in the architecture and design of the system. Using performance testing tools, we can begin practicing risk management.

Load testing and stress testing are merely subsets of performance testing. As such, performance testing is a wide term that encompasses:

- Spike testing
- Endurance testing
- Volume testing
- Scalability testing

How frequently you test performance depends on your development methodology:

- A team using a waterfall approach will want to test with each individual release.
- If they are using the agile methodology, performance testing should be a constant practice alongside development.

Why use performance testing?

Want to establish the benchmark behavior of your system? The first step to testing your functionality is performance testing. During performance testing, you're looking to meet or exceed a number of industry-defined benchmarks.

Importantly, performance testing is not in any way aimed at finding defects with the application.

Rather, it's meant to set the benchmark and standard for the application. The key is to ensure that utmost accuracy is observed during performance testing. Close monitoring is necessary for ensuring the performance of a system or application.

When setting a benchmark, examine attributes such as:

- Throughput
- Response time
- Speed
- Stability
- Resource usage

These are examples of KPIs—Key Performance Indicators. Essentially, performance testing analyzes KPIs like these.

Performance testing best practices

When or how often to performance test depends on what kind of testing you are doing. Here are some tips for getting the most out of your performance testing measures.

Your performance testing should not include any actual users. Performance testing should use software to measure how well it performs or does not perform. User Acceptance Testing (UAT) is when we start to use real software or system users to see how it performs in the real world.

Test as often as possible. There are specific times to test, usually near the last step. When it is time to test, do it repeatedly. Repeat tests over and over to get the most accurate results. If your application, software, or website can successfully pass the tests over and over again, you can tick it off as ready for the next stage.

Do not change what you are testing. When we make lots of changes to an experiment, it becomes more difficult to identify what exactly we're testing. Between each test, your environment should stay the same. This includes the load or stress that you place upon the system.

Use automated tools to make the process as simple as possible. Automating your testing process will boost efficiency. You can:

- Save time and money
- Minimize navigating complex architecture
- Provide more accurate results faster

Common tools for performance testing include:

- Apache JMeter
- Ready API
- BlazeMeter
- Visual Studio Test Professional
- Micro Focus LoadRunner

What is load testing?

Load testing is the process of putting pressure on a software, system, or device to measure its response under an expected load.

By simulating production (pressure), load testing shows the behavior under normal and *expected* peak conditions. The goal is to ensure a given function, system, or program can handle what it's designed for. This is important because when you're building your product, you're only accounting for a few individual users. Load testing helps you prepare for what happens when you deploy your product to hundreds or thousands of users.

More specifically, the goals of load testing include:

- **Determining the upper limit of all the components of an application** (the hardware, database, network, etc.). This test ensures that the application has the ability to manage any load it will be exposed to in the future.
- **Detecting defects in applications** that relate to buffer overflow, memory mismanagement, and memory leaks. Load testing is likely to expose a number of issues including bandwidth issues, load balancing problems, and the carrying capacity of the system.

Usually, you load test a system that is as close as possible to becoming a finished product, ready for deployment to the masses. It shows your team whether the product is working as expected and/or intended, under the given conditions.

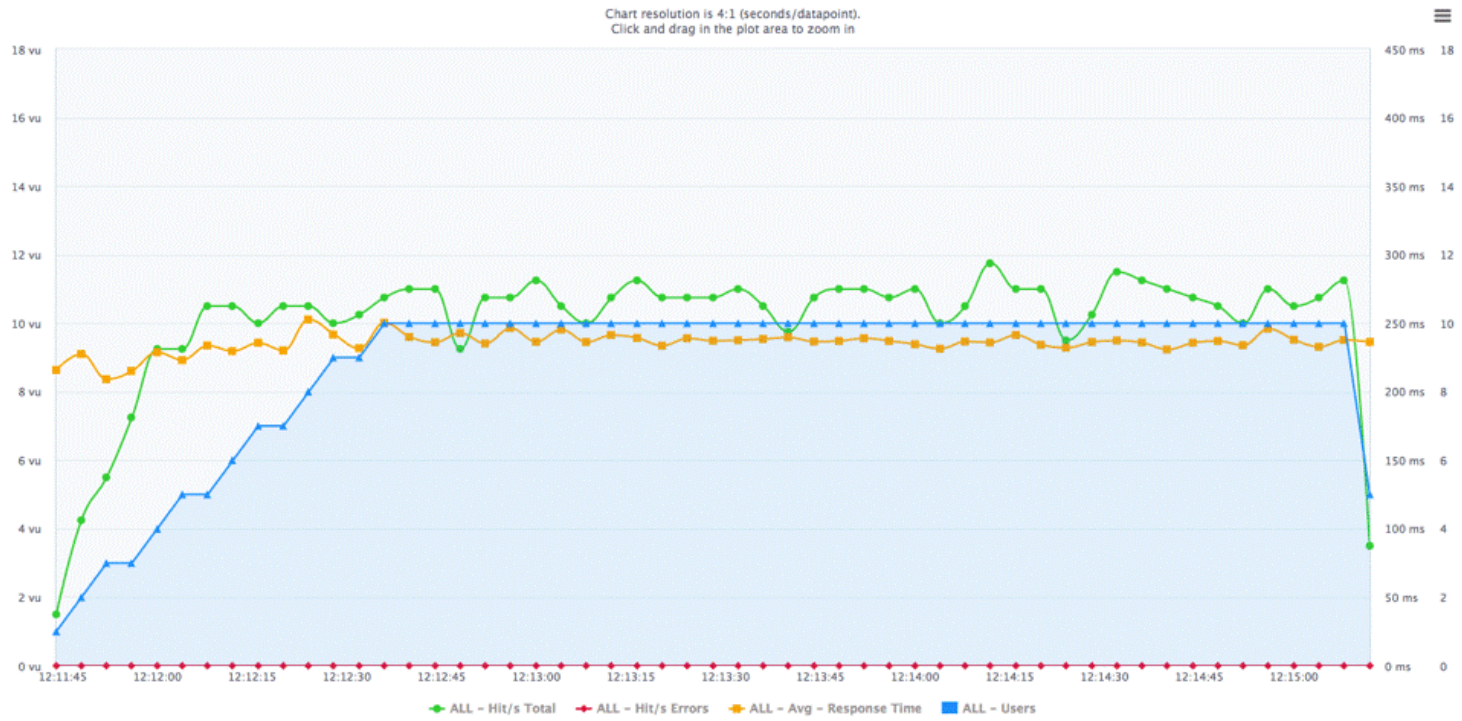
How load testing works

During load testing, the system is taken through a steadily and constantly increasing load until it reaches the threshold. You want to find the weight limit—the weight (load) of activity that an application, software, or website can handle under normal conditions.

The weight limit should be our peak load. Load testing should help you identify:

- Page load issues
- System lag
- Additional issues arising when multiple users accessing the system

Good load testing probably looks something like this:



(Source)

We have four bars to look at:

- Green means hits (visitors)
- Red means errors
- Yellow means response time
- Blue means users

This is a good experiment for a number of reasons. We want to see the blue and green lines climbing while the yellow one stays relatively flat. If the yellow line drops, we know that the software can't stand up to high demands (users and visitors).

As we add hits to the software, the system should be able to withstand traffic until we reach our expected weight limit. Seeing a drop in the yellow line on the graph would indicate that there is a problem and that the weight limit is a lot lower than desired. If it can't perform well with the expected load, there may be some serious problems in the design.

Of course, you want to know what will happen to the system when the traffic increases suddenly. You may not be able to detect some of these issues during development and testing. This could include testing for:

- Traffic from different or varying geo-locations
- Sudden spikes in traffic

When to load test

Load testing as a process doesn't start till a project is close to its end. This is when system performance and actual user engagement can undergo accurate simulation and testing.

(Technically, you don't have to wait till software development is completed before testing. You can test a specific component in a focused way for the load at any stage of the development process.

Still, to test the demand of hundreds of thousands of users, it's best to wait till development is nearly complete.)

Load testing tools usually mimic the actions of multiple concurrent users of the program, website, app, etc. It is a repeated procedure that involves collecting and monitoring both software and hardware statistics. Of particular interest are the:

- CPU
- Disk IO
- Memory of the physical servers

The test will also show the response time and throughput of the system along with other KPIs. When checking backend performance issues, the endurance of input over prolonged periods of time or simultaneous user input is important to understand. However, checking for things that could cause lag, broken functionality, or memory leaks before completing development is load testing in a limited form.

After analysis, the load testing software will generate a load testing report. Compare these results to your expected goals (benchmarks).

What is stress testing?

Where load testing tests expected loads, stress testing applies unrealistic load scenarios—to the point of overloading the system till it breaks. The aim is to find out how stable your product is by stretching it beyond its bandwidth capability. Stress testing evaluates how an application will behave beyond normal conditions and normal peak load.

Why do you need to know this ahead of time? The biggest reason is to see whether your sensitive data remains protected when the system fails. Is there a particular situation where you're more vulnerable to data leaks?

When you stress test, you deliberately induce failures to analyze the risks at the breaking points. You can then tweak the programs to make inevitable failures more graceful. If it's going to break, we want our functionality testing to show that it can recover! Exploring the outer limits of a system is the best way to do this.

Some of the issues that can be exposed during stress testing include:

- Memory leaks
- Race conditions

- Synchronization issues

When you've completed a round of stress testing, analyze the post-crash report to understand whether your application failed gracefully, as expected—or not.

How to stress test

Stress testing works on anything you're developing: software, an app, a website, or a backend system, like a host server. The key to successful stress testing is ensuring a controlled environment before and during the test.

Stress testing can include various activities such as:

- Overloading the system with lots of jobs
- Removing system components
- Checking the behavior when the number of users suddenly increases (known as spike tests)
- Checking the sustainability of a system over a period of time through a gradual increase in the number of users (known as soak tests)

Performance testing is critical

Performance testing of any kind is not merely a development need—it's a business requirement.

Stakeholders and customers demand fast and reliable digital experiences. This means that your applications must be able to support thousands or even hundreds of thousands of users seamlessly. Knowing what test to do at what time and for what purpose will help you achieve this goal.

Still confused? This is a helpful video about JMeter, an open source performance testing tool:

Related reading

- BMC IT Operations Blog
- BMC DevOps Blog
- Shift Left Testing: What, Why & How To Shift Left
- Testing in DevOps: Introduction & Best Practices
- What's Testing as a Service? TaaS Explained
- Beginner's Guide to Test Automation Frameworks