



Key Considerations in Performance Testing

Leslie Segal
President
Testware Associates, Inc.

Testware Associates, Inc.

242 Old New Brunswick Rd., Suite 240, Piscataway, NJ 08854
tel: 732-562-0155 email: info@testwareinc.com Web: www.testwareinc.com

Contents

<i>Introduction</i>	<i>1</i>
<i>Performance Testing Vs. Functional Testing</i>	<i>1</i>
<i>“Fluid” Pass/Fail Criteria</i>	<i>1</i>
<i>Common Performance Testing Objectives</i>	<i>2</i>
<i>Key Performance Metrics</i>	<i>2</i>
<i>Helpful Data To Gather During Performance Testing</i>	<i>2</i>
<i>Performance Degradation</i>	<i>3</i>
<i>Creating Performance Test Scenarios</i>	<i>3</i>
<i>Real vs. Virtual Users</i>	<i>4</i>
<i>The Pitfalls of Misunderstanding Virtual Users</i>	<i>4</i>
<i>Example – Performance Testing an E-Commerce Website</i>	<i>5</i>
<i>Summary</i>	<i>6</i>
<i>About Testware Associates</i>	<i>7</i>
<i>References</i>	<i>7</i>

Introduction

To accomplish meaningful performance tests, the use of automated performance testing tools is required. If these tools are used improperly, or the “scripts” that implement the tests are written incorrectly, the test results may lead you to think your Web site can support 1,000 concurrent users, when, in reality, 10 users executing a particular function cripple the site. Or, you might conclude the opposite – that your system can support only 10 users, when it can really support 1,000 – and invest unnecessarily in additional hardware to increase capacity.

Performance testing requires a different approach and a different skill set from those used in executing functional and system testing. This paper will present an overview of performance testing, with emphasis on topics that Testware deem “key considerations” based on our years of experience in testing.

Performance Testing Vs. Functional Testing

Let’s start by comparing performance testing to functional testing. Functional testing is a fairly easy concept to grasp and the results are concrete. A functional test either passes or fails; very black and white. The function works according to specifications, or it doesn’t. Performance testing, on the other hand, is neither concrete nor black and white. What does “passing” mean for a performance test? Does it mean that if 300 users are simultaneously accessing a website and the average response time is less than 10 seconds, the test passes? What if the average response time is 10 seconds, the maximum response time is 90 seconds and the standard deviation is 35 seconds? Does the test *still* pass? For some systems, passing simply means that with a large number of users on the system, the system doesn’t crash. One of the most difficult tasks in performance testing is to define what you want to measure and the pass/fail criteria.

“Fluid” Pass/Fail Criteria

We sat down with one client to discuss exactly what their pass/fail criteria were. This client had recently read an article that claimed if a Web page takes more than 5 seconds to load, users would abandon the site and go to a competitor’s site. Based on this article, our client asked us to measure the time it took to load each individual page on their site, varying the user load from 1 to 500 users. The test passed if the load time for each page was, on average, 5 seconds or less for the entire range of user loading (1 to 500 users). Our tests yielded an average response time ranging from a low of 20 seconds to a high of 90 seconds across the individual pages; none of the pages met the 5 seconds or less “pass” criterion. Upon presenting this data to our client, they decided that this 20-90 second average response time range was fine and that all of the tests should be passed.

There is a fairly low percentage of applications that have ‘real’ performance requirements. ETrade® used to guarantee that any trade placed would be executed within 30 seconds. Their website now reads, “System response and account access times may vary due to a variety of factors, including trading volumes, market conditions, system performance, and other factors.”

More often than not, your site’s performance should be equal to or better than your competitors’ performance. If a competitor’s Web pages load in 10 seconds, your Web pages typically should load in 10 seconds or less.

Common Performance Testing Objectives

Determining the measurements and criteria for the performance tests is easier once you determine what your performance testing objectives are. Some of the more common objectives are:

- Find existing bottlenecks
- Verify scalability of the system
- Verify impact of new features/functions
- Verify current system capacity
- Verify performance requirements
- Determine optimal hardware/application configuration

Unless you have all the time and money in the world, it usually doesn't make sense to address *all* of these objectives at the same time. If you tried, chances are that by the time you get through all of these objectives, your application will have gone through many major changes over several releases.

Where's a good place to start? Finding existing bottlenecks. All systems have bottlenecks in them, each affecting / limiting performance in a certain way. Once you eliminate a bottleneck, there will be another one right behind it. Current system capacity and system scalability are often directly related to a bottleneck.

Key Performance Metrics

The performance of your system should be measured in, at least, these three areas:

- Response time: how long does it take the system to respond to a request for a piece of information
- Throughput: how much activity can the system support, often measured in transactions or hits/second
- Round trip time: how long does the entire user-requested transaction take, including connection and processing time

Creating and executing scripts using a performance testing tool is rather easy to do, all things considered. The hard part is figuring out what scenarios to implement with the tool and then what to do with the large amount of data the tool generates. In the end, you certainly want to reach the right conclusions about the true performance of your system!

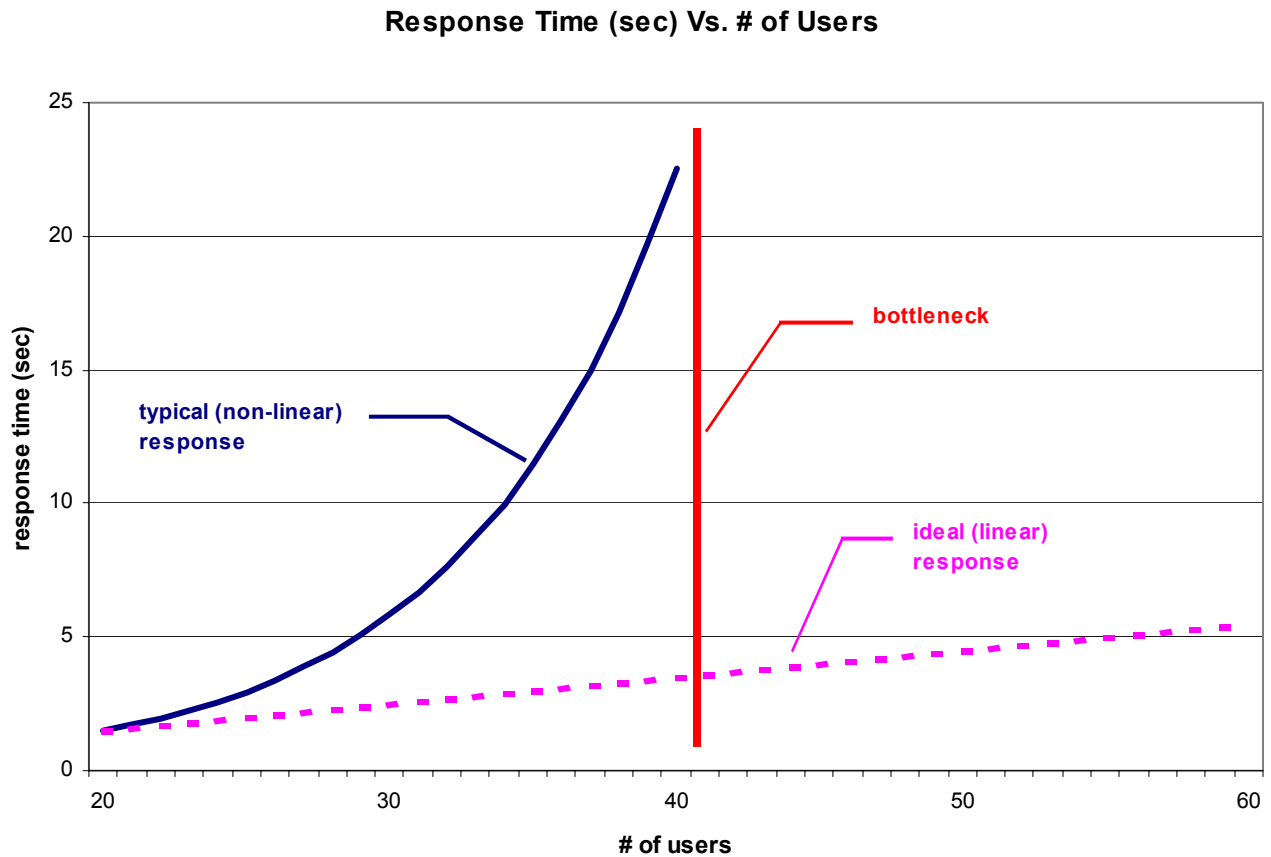
Helpful Data To Gather During Performance Testing

While executing performance tests, it is important to collect metrics on what your application is doing. It is not enough to simply report that when 100 users were on the system, it crashed. You can help the development team by recording server (Web, database, application) resource usage and correlating that data to the data captured by the performance testing tool. Common statistics to record include processor usage, memory usage, disk I/O rates and network traffic. Then, instead of reporting only that the system crashed at 100 users, you might also be reporting that the application server was out of physical memory and its processor was at 75% utilization just before the crash.

Performance Degradation

One of the common misconceptions about the performance of a system is that as you add users to the system, performance degrades gracefully in a linear fashion, slowly but ultimately arriving at a point where response time is unacceptable. See the line labeled “ideal (linear) response” in the graph below.

In most cases, things are a lot less ideal. See the line labeled “typical (non-linear) response” in the graph. As soon as we approach a bottleneck, performance degrades drastically and often the system just hangs or crashes after this bottleneck is reached.



Creating Performance Test Scenarios

Knowledge of how users currently or will use your system is a “must have” when determining what scenarios will be performance tested. It doesn’t make sense to test every possible scenario, and there wouldn’t be enough time and money to do that anyway.

If your system is already live, perhaps you are gathering statistics on how your customers are using it. For example, you might be using a tool like WebTrends® to capture and track user activity on your website. If your system is brand-new, you’ll need to design your performance test scenarios based on user behaviors in similar environments.

Consider an e-commerce website. The simplest of scenarios is loading the site's home page. A more complicated scenario is loading the home page, then accessing the search page, searching for a particular product, and adding that product to the shopping cart.

Once the scenarios are defined, it is necessary to determine

- How many times that scenario should be executed over a given period of time (1 hour, several hours, 1 day, 1 week, etc.)
- How many users should be executing the scenario concurrently (for example, 100)
- How the users' concurrent scenario execution is timed. Do the 100 users execute the scenario at exactly the same instant, are they spaced linearly over a period of 5 minutes, or something else?

Real vs. Virtual Users

If you are the sole visitor currently accessing www.testwareinc.com, one real user (you) is accessing the site. If 99 of your colleagues join you in browsing around www.testwareinc.com, then 100 real users are accessing the site.

Performance testing tools revolve around "virtual users." Performance testing tools run on high-powered PCs (or other hardware) that cause the tool's virtual users to generate the traffic and activity associated with tens, hundreds or thousands of real users. Scripts are written in a tool-specific language that cause the performance tool's virtual users to access the website under test (or client/server application, etc.) and emulate the activities real users would engage in.

Performance test tool pricing is proportional to the number of virtual users that you license/buy. The cost of a license for 250 virtual users is significantly less than one for 2,500 virtual users. So, if you don't really need 2,500 virtual users, why spend the additional money on both the tool and all of the hardware that you will need to support running the additional virtual users? For example, you'll typically need about one PC for each 250-500 virtual users being tested. That's one PC for 500 virtual users and ten PCs for 5,000 users. Big difference!

The Pitfalls of Misunderstanding Virtual Users

Some companies assume that the relationship of real users to virtual users is always 1:1, meaning that to test 1,000 real users (for example) on their site, a 1,000 virtual user license is required. This isn't always the case; depending on what has to be tested and how you script it, it's possible to have a single virtual user act as multiple real users. We recently implemented a performance test where the load 500 real users would generate was tested with just 25 virtual users. A 25 virtual user license is certainly cheaper than a 500 virtual user license; recognizing this 20:1 real to virtual user ratio allowed us to save this client money. Knowing the right ratio of real to virtual users will save you money; you'll avoid "overbuying" virtual users. Determining this ratio requires knowledge of actual or predicted application usage/traffic patterns, well-defined performance testing goals and a fair bit of mathematics. (The details of such a determination are beyond the scope of this paper.)

Let's consider another danger of misunderstanding virtual users. Assume that a company needs to test 5,000 users accessing their website and mistakenly believes the right ratio of real to virtual users is 1:1. That company gets pricing for a 5,000 virtual user performance testing license, but should really be looking at a 500 user license. Based on the 5,000 virtual user pricing, the company decides that performance testing is too expensive and decides to launch the site without it, keep their fingers crossed and hope the performance is acceptable. In addition to saving money, understanding your real

users and their correlation to virtual users will allow you to generate more accurate scripts and in turn, more precise results.

Example – Performance Testing an E-Commerce Website

In order to define performance testing scenarios, it is often helpful to create a simplified model of the system under test. This helps you to understand the relationships of the entities in the application.

Let's consider an e-commerce website where users can log in, look around and/or buy products while logged in, and log out. While logged in, users can toggle between looking and buying.

Now, let's define the profiles of some real users of this system:

Real User #1 is Sally Surfer. Sally spends about two hours on the site daily, mostly just looking at items. The probability that she will purchase anything in her two-hour session is about 20%.

Real User #2 is Buying Betty. Betty spends about 20 minutes on the system every day. Since Betty just loves shopping, her probability of a purchase is 85%. Of course her probability of returning an item is about 50%, but since returns aren't processed through the Web site, it doesn't impact our testing.

Real User #3 is Thinking Tony. Tony will spend about one hour a day on the site, usually just looking at the same page and thinking about buying the product featured on that page. The probability that he will make a purchase during a session is about 10%.

Real User #4 is Quick Charlie. Charlie doesn't like to waste time; he goes on the site knowing what he wants. If it's there, he'll buy it. If not he logs off. He will spend an average of about five minutes on the site and 50% of the time he will make a purchase.

Since the overall activity level from these four users doesn't add up to much, we have two options:

1. Create a separate scenario for each of the four users, and use a total of 4 virtual users to performance test – one virtual user per scenario.
2. Create a single scenario that causes the same activity on the Web server that the four separate scenarios in the previous option do. Execute this single scenario with a single virtual user.

Which is the right option? It depends. If you need to test how the system handles four users logging on at precisely the same instant in time, then you need four virtual users. If your goal is to measure the performance of the system under the load normally generated by the four users named above, then a single scenario executed by one virtual user would work. Having four users log in at *exactly* the same point in time is not the same as showing that your application can handle four users logging in over a five-minute period of time.

The relationship between the real users of your application and the virtual users needed to emulate the corresponding load is very much dependent upon the behavior of the application itself. If your application generates a large volume of traffic (a high number of hits/second on the Web server) or a great deal of processing time is required by certain tasks, the relationship of real to virtual users may be one-to-one. On the other hand, if response time for individual transactions is fairly small and the system requires significant user "think time," the relationship may be 10:1, 100:1 or even higher.

Summary

Once you've

- Determined your performance testing objectives
- Set your pass/fail criteria
- Established the profiles of your real users
- Determined the proper ratio of real to virtual users
- Created your test scripts and executed them
- Gathered and correlated the data from the test tool and Web / application / other servers

you are in the position to draw conclusions from all of that data and judge if the tests passed or failed.

Where can things go wrong?

- If you've misjudged the ratio of real to virtual users, you might conclude that performance testing is too expensive and decide to skip performance testing altogether.
- If you test and incorrectly conclude that your system's performance is inadequate, you will spend money unnecessarily to fix a performance problem that doesn't exist, by buying additional hardware or attempting to optimize code.
- Arguably, the worst outcome is to test and conclude that your system's performance is adequate when it is not; the performance problems and possible crashes / hang-ups in your system will be there for users to find.

About Testware Associates

Testware Associates, Inc. is a testing services firm that helps clients address their “testing pain” and reduce the risks and costs associated with poor quality software. Testware serves clients by performing the entire end-to-end testing function or by handling selected parts of the overall test effort. Testing work is performed at the client’s site, at Testware’s in-house testing lab or a combination of the two. It’s never too early or too late for Testware to become involved – whether you are building a brand-new, leading edge system or periodically updating a live system, Testware can help! Please visit www.testwareinc.com for more information.

References

The Practical Performance Analyst, Neil Gunther, McGraw-Hill, 1998.

Web Server Technology, Nancy Yeager and Robert McGrath, Morgan Kaufmann Publishers, 1996