



Your Results. Our Commitment.

Approach To Performance Testing

Prepared by

3Nexus LLC



Table of Contents

1.0 INTRODUCTION	2
2.0 OVERVIEW OF PERFORMANCE TESTING.....	2
2.1 REPEATABILITY.....	2
2.2 PERFORMANCE TESTS VERSUS LOAD TESTS	2
2.3 PRE-REQUISITES FOR PERFORMANCE TESTING	3
2.4 TYPES OF PERFORMANCE TESTING	4
2.5 REPORTING METRICS	4
2.6 KEYS TO ACCURACY	5
2.7 LOAD SCALABILITY.....	5
3.0 PROCESS STEPS	6
3.1 CONDUCTING A SYSTEM ANALYSIS.....	6
3.2 USAGE PATTERNS UNDERSTANDING THE NATURE OF THE LOAD	6
3.3 USAGE PATTERNS USER SESSION VARIABLES / PAGE REQUEST DISTRIBUTION	6
3.4 DETERMINE RANGE & DISTRIBUTION OF VALUES FOR THESE VARIABLES	6
3.5 ESTIMATING TARGET LOAD LEVELS	7
3.6 DEVELOP BASE TEST SCRIPTS LOAD TEST DESIGN	7
3.7 DETERMINE FREQUENCY OF EXECUTION	7
3.8 EXECUTE TEST SCRIPTS.....	7
3.9 MONITOR EXECUTION.....	8
3.10 COLLECT & EXAMINE REPORTS.....	8
3.11 PROVIDE REPORTS AND SUGGESTIONS	8
4.0 TOOLS USED.....	9
4.1 WEBLOAD	9



1.0 Introduction

3Nexus provides offshore and on-site quality assurance services. Our dedicated staff of QA professionals to perform test plan creation, script development for automated testing, manual and automated test plan execution, and stress, load and capacity testing.

As part of our suggested QA guidelines, 3Nexus encourages and supports stress, load and performance testing. The elements contained in these tests are critical to the successful sustainability of any client/server applications and web-based applications.

2.0 Overview Of Performance Testing

Performance Tests determine end-to-end timing (benchmarking) of various time critical business processes and transactions, while the system is under low load but with a production sized database. This sets 'best possible' performance expectation under a given configuration of infrastructure. It also highlights very early in the testing process if changes need to be made before load testing should be undertaken.

For example, a customer search may take 15 seconds in a full sized database if indexes had not been applied correctly, or if an SQL 'hint' was incorporated in a statement that had been optimized with a much smaller database. Such performance testing would highlight such a slow customer search transaction, which could be re-mediated prior to a full end to end load test.

It is 'best practice' to develop performance tests so that response times from a user perspective can be measured in a repeatable manner with a high degree of precision.

2.1 Repeatability

A key indicator of the quality of a performance test is repeatability. Re-executing a performance test multiple times should give the same set of results each time. If the results are not the same each time, then the differences in results from one run to the next cannot be attributed to changes in the application, configuration or environment.

2.2 Performance Tests versus Load Tests

The best time to execute performance tests is at the earliest opportunity after the content of a detailed load test plan have been determined. Developing performance test scripts at such an early stage provides opportunity to identify and re-mediate serious performance problems and expectations before load testing commences.

Load testing is done by creating a set of virtual users that simulate a load on a real application. It increases confidence and knowledge of response times, performance, reliability and scalability. Without conducting proper load tests, it is difficult or impossible to know the number of users an application can effectively support. Load testing increases up time, customer satisfaction, and decreases both real and potential costs of hardware, software, and development resources. Load testing can identify bottlenecks and defects that cannot be found with conventional methods of testing.

For example, management expectation of response time for a new web system that replaces a block mode terminal application is often articulated as 'sub second'. However, a web system, in a single screen, may perform the business logic of several legacy transactions and may take 2 seconds. Rather than waiting until the end of a load test cycle to inform the stakeholders that the test failed to meet their formally stated expectations, a little education up front may be in order. Performance tests provide a means for this education.

Another key benefit of performance testing early in the load testing process is the opportunity to fix serious performance problems before even commencing load testing.

A common example is one or more missing indexes. When performance testing of a "customer search" screen yields response times of more than ten seconds, there may well be a missing index, or poorly constructed SQL statement. By raising such issues prior to commencing formal load testing, developers and database administrators can check that indexes have been set up properly.

Performance problems that relate to size of data transmissions also surface in performance tests when low bandwidth connections are used. For example, some data, such as images and "terms and conditions" text are not optimized for transmission over slow links.

2.3 Pre-requisites for Performance Testing

A performance test is not valid until the data in the system under test is realistic and the software and configuration is production like. The following table list pre-requisites for valid performance testing, along with tests that can be conducted before the pre-requisites are satisfied:

Performance Test Pre-Requisites	Comment	Caveats on testing where pre-requisites not satisfied
Production Like Environment	Performance tests need to be executed on the same specification equipment as production if the results are to have integrity.	Lightweight transactions that do not require significant processing can be tested, but only substantial deviations from expected transaction response times should be reported. Low bandwidth performance testing of high bandwidth transactions where communications processing contributes to most of the response time can be tested.
Production Like Configuration	Configuration of each component needs to be production like. For example: Database configuration and Operating System Configuration.	While system configuration will have less impact on performance testing than load testing, only substantial deviations from expected transaction response times should be reported.
Production Like Version	The version of software to be tested should closely resemble the version to be used in Production.	Only major performance problems such as missing indexes and excessive communications should be reported with a version substantially different from the proposed production version.
Production Like Access	If clients will access the system over a WAN, dial-up modem, DSL, ISDN, etc. Then testing should be conducted using each communication access method.	Only tests using production like access are valid.
Production Like Data	All relevant tables in the database need to be populated with a production like quantity with a realistic mix of data. For example: Having one million customers, 999,997 of which have the name "John Smith" would produce some very unrealistic responses to customer search transactions.	Low bandwidth performance testing of high bandwidth transactions where communications processing contributes to most of the response time can be tested.



2.4 Types of Performance Testing

Listed below are some of the types of performance testing methods that we employ. This is followed by the metrics we capture in this process. These metrics allow us to gauge how the system will stand up to high volume of transactions/hits in real time. Any bottlenecks that may exist in the application will be exposed in this process and remedial measures can be taken. Such an exercise will ensure that there are no unexpected surprises after a system/application goes live.

2.4.1 Load Testing

Testing an application against a requested number of users. The objective of this test is to determine whether the site can sustain this requested number of users with acceptable response times.

2.4.2 Stress Testing

Load testing over an extended period. The objective is to validate an application's **stability and reliability**.

2.4.3 Capacity Testing

Testing to determine the maximum number of concurrent users an application can manage. The objective is to benchmark the maximum loads of concurrent users a site can sustain before experiencing system failure.

2.5 Reporting Metrics

Below is a listing and explanation of some of the key metrics that we capture when we conduct performance testing on a system / application.

2.5.1 Load size

The number of concurrent Virtual Clients trying to access the site.

2.5.2 Throughput

The average number of bytes per second transmitted from the ABT (Application being tested) to the Virtual Clients running this Agenda during the last reporting interval.

2.5.3 Round Time

It is the average time that it took the virtual clients to finish one complete iteration of the agenda during the last reporting interval.

2.5.4 Transaction Time

The time it takes to complete a successful request, in seconds. (For a web-based application, each request for each gif, jpeg, html file, etc. is a single transaction.) The time of a transaction is the sum of the Connect Time, Send Time, Response Time and Process Time.

2.5.5 Connect Time

The Time it takes for a Virtual client to connect to the Application Being Tested.

2.5.6 Send Time

The time it takes (in seconds) the Virtual Clients to write a request to the Application Being Tested (ABT).



2.5.7 Response Time

The time it takes (in seconds) the ABT to send the object of a request back to a Virtual Client. In other words, the time from the end of the request until the Virtual Client has received the complete item it requested.

2.5.8 Process Time

The time it takes to parse a response from the ABT and then populate the Document Object Model (DOM), in seconds.

2.5.9 Wait Time (Average Latency)

The time it takes from when a request is sent until the first byte is received.

2.5.10 Receive Time

The elapsed time between receiving the first byte and the last byte.

2.6 Keys to Accuracy

- ◆ Recording ability against a real client application
- ◆ Capturing protocol-level communication between client application and rest of system
- ◆ Providing flexibility and ability to define user behavior
- ◆ Verifying that all requested content returns to the browser to ensure a successful transaction
- ◆ Showing detailed performance results that can easily be understood and analyzed to quickly pinpoint the root cause of problems
- ◆ Measuring end-to-end response time
- ◆ Using real-life data
- ◆ Synchronizing virtual user to generate peak loads
- ◆ Monitoring different tiers of the system with minimal intrusion

2.7 Load Scalability

- ◆ Generating the maximum number of virtual users that can be run on a single machine before exceeding the machine's capacity
- ◆ Generating the maximum number of hits per second against a Web/application server
- ◆ Managing thousands of virtual users
- ◆ Increasing the number of virtual users in a controlled fashion

3.0 Process Steps

3.1 Conducting a System Analysis

During this phase, the entire system is analyzed and broken down into specific components. Any one component can have a dramatic effect on the performance of the system. This step is critical to simulating and understanding the load and potential problem areas. Furthermore, it can later aid in making suggestions on how to resolve bottlenecks and improve performance. Some example tasks include:

- ◆ Identify all hardware components
- ◆ Identify all software components
- ◆ Identify all unrelated processes and services that may affect system components

3.2 Usage Patterns Understanding the Nature of the Load

Understanding from where and when the load comes is necessary to correctly simulate the load. With this information, minimum, maximum, and average loads can be determined. The distribution of the load can also be determined.. Some data that is collected includes:

- ◆ Visitors' IP addresses
- ◆ Date & time each page visited
- ◆ Size of the file or page accessed
- ◆ Code indicating access successful or failed
- ◆ Visitor's browser & OS / client-side configuration

3.3 Usage Patterns User Session Variables / Page Request Distribution

What processes that the load triggers is needed to simulate the correct load? Information that is collected is as follows:

- ◆ Length of the session (measured in pages)
- ◆ Duration of the session (measured in minutes and seconds)
- ◆ Type of pages that were visited during the session, and what processes are triggered for each

3.4 Determine Range & Distribution of values for these variables

With the previous two phases, statistical data can be formed. Some distribution figures that are computer are the following:

- ◆ Average (for example, 4 page views per session)
- ◆ Standard deviation
- ◆ Discrete distribution

3.5 Estimating Target Load Levels

With the guidance of the client, and reflecting on the data previously collected, target load levels, or goals, can be established. Such goals often include:

- ◆ Overall traffic growth (historic data vs. sales/marketing expectations)
- ◆ Peak Load Level (day of week, time of day, or after specific event)
- ◆ How long the peak load level is expected to last
- ◆ Options on getting these numbers: Concurrent users, Page views, User session per unit of time, etc.

3.6 Develop Base Test Scripts Load Test Design

During this phase the base scripts that will be used to generate the load are determined. The steps and some examples follow.

3.6.1 Test Objective

SAMPLE: The objective of this load test is to determine if the Web site, as currently configured, will be able to handle a predefined number of sessions/hr-peak load level anticipated for the coming holiday season. If the system fails to scale as anticipated, the results will be analyzed to identify the bottlenecks, and the test will be run again after the suspected bottlenecks have been addressed.

3.6.2 Pass / Fail Criteria

SAMPLE: The load test will be considered a success if the Web site will handle the target load of sessions/hr while maintaining the average page response times defined below. The page response time will be measured over T1 lines and will represent the elapsed time between a page request and the time the last byte is received.

3.6.3 Script Labels and Distribution

- ◆ Type of scripts (keeping Page Request Distribution as the main concept here)
- ◆ Number of Scripts (usually 10 -15)
- ◆ Naming of Scripts
- ◆ Calculate the target page distribution based on the types of pages hit by each script, and the relative frequency with which each script will be executed.

3.7 Determine Frequency of Execution

When, and how often the load test is run can have an affect on the outcome of the results. Other processes and unrelated loads may impact the performance of the application under test. During this stage, the proper tests are determined to run at specific points, or specific number of iterations.

3.8 Execute Test Scripts

- ◆ Combine Scripts to create load-testing Scenario
- ◆ Scripts Executed
- ◆ Percentages in which those scripts are run
- ◆ Description of how load will be ramped up



3.9 Monitor Execution

During execution, systems will be automatically logged and monitored. At times, systems may also be manually monitored for immediate problems. Data gathered in this phase will be used to generate reports that will be used to judge the performance of the system and each system component.

3.10 Collect & Examine Reports

3Nexus utilizes data collected from WebLOAD, LoadRunner or other customized performance testing tools to provide the client with easy to use, relevant information. Given below are some of the reporting metrics available that can be derived from the data collected.

Report on	Details
Transaction Summary	This indicates the number of transactions that passed, failed, aborted or ended with errors.
Throughput	This indicates the amount of throughput (in bytes) on the Web server during load testing
Rendezvous	Indicates when and how virtual users are released at each point
Transactions / second (Passed)	The number of completed, successful transactions performed per second
Transactions / second (Failed)	The number of incomplete, failed transactions per second
Percentile	Analyzes percentage of transactions that were performed within a given time range
Performance Under Load	Transaction times relative to the number of virtual users running at any given point during the scenario
Transaction Performance	Average time taken to perform transactions during each second of scenario
Transaction Performance Summary	Minimum, maximum, and average performance times for all transactions in scenario
Transaction Performance by Virtual User	Time taken by an individual virtual user to perform transactions during the scenario
Transaction Distribution	The distribution of the time taken to perform a transaction
Connections per Second	Shows the number of connections made to the Web server by virtual users during each second of the scenario run

3.11 Provide Reports and Suggestions

Summary reports of the performance tests will be provided with respect to the metrics above, or the metrics the customer requires. After analyzing these reports, 3Nexus will recommend courses of action to resolve bottlenecks, increase scalability, and performance.



4.0 Tools Used

A number of tools are available for generating performance tests on web-based and client-server applications. However, some applications require the use of custom developed tools. 3Nexus can develop a custom tool if the situation warrants. Below is given a brief description of the capabilities and system requirements of one of the typical tools we use WebLOAD.

4.1 WebLOAD

WebLOAD simulates the volume and configuration of users you need, while maintaining the smallest hardware configuration. WebLOAD maximizes your resources by minimizing the size of the footprint a virtual user consumes so you get more load from your hardware than you would with any other performance tool. While maintaining an efficient footprint, WebLOAD also scales to Olympic proportions when your applications demand it, enabling you to meet the challenges of peak -demand and high-volume tests.

With WebLOAD you can generate load and simulate traffic in both .NET and J2EE Environments, and from your choice of platform, or any combination of platforms allowing you to leverage the infrastructure you already own or have easy access to. Solaris, Linux, WinTelyou choose, the same script will run on all of them.

4.1.1 Scalable Client-side JavaScript Support

Execute client-side browser activity in a scalable and portable manner, so you can accurately emulate real user activities on a modern browser effectively and efficiently from any load platform.

- ◆ Full statistics support
- ◆ Full debugging capabilities
- ◆ Recovery mechanism for robust test scripts

4.1.2 Visual Agenda Authoring Tool (VAAT)

WebLOAD provides a simple to use yet powerful visual environment for creating test scripts. For novice users point and click to automatically record test scripts that you can play back and preview using your browser. For advanced users jump directly to JavaScript (our standards-based scripting language) to create your test scripts.

- ◆ Full JavaScript debugger and new JavaScript editor
- ◆ Form data wizard
- ◆ Integrated browser playback
- ◆ Same authoring environments for all RadView products offers full compatibility

4.1.3 Web Application Tools Integration

Out-of-the-box performance measures and reports for the most common application tools.

- ◆ IBM's WebSphere
- ◆ Sun's SunOne
- ◆ BEA's WebLogic
- ◆ Plus Apache, Oracle, SQL Server, IIS, ASP, Real, UC-Davis, Windows, and more



4.1.4 Script Sharing

One can leverage the same test scripts without modification in WebFT.

4.1.5 Web Services Testing

Easy-to-use wizards automate the process of testing non-GUI-based Web Services.

- ◆ Test an application at the API (application interface) and Unit level
- ◆ Support for emerging technologies (.NET, SOAP, WSDL, BPELW45)

4.1.6 Notification Manager

Automated alerts notify administrators via email or pager when application performance falls outside of a user-defined threshold. By monitoring response time, hits per second, failed connections and many other variables, users can receive instant notification when a problem is encountered.

- ◆ Alerts available on over 35 performance metrics
- ◆ Easy set-up via dialog box
- ◆ Integrates with any SMTP mail server, including authenticated logins

4.1.7 Open Standards Support

- ◆ Open standards JavaScript scripting language
- ◆ Internet Technology Support - JavaScript, XML, Java, EJB, ActiveX, WAP, HTTP, SNMP, Real and Microsoft Streaming Technologies

4.1.8 Real-World Simulation

- ◆ Measure performance under any real-world scenarios including traffic patterns, changes to activity, peak load, computing environment, and protocols, to name a few of the virtually unlimited variables existing in today's demanding Internet environment.
- ◆ Define a variety of Virtual Clients characteristics such as connection speed, browser type, multi-threading, SSL encryption strength, etc. Configure a mix of these Virtual Client types to execute the load test simultaneously or in random combinations for accurate real-world simulation.
- ◆ WebLOAD automatically handles dynamic Session ID's with its Automated State Management and provides platform independent native access to the Document Object Model (DOM), to provide more intelligent information about the structural elements of your Web application.

4.1.9 Enterprise Performance, Scalability and Resource Efficiency

- ◆ WebLOAD sets new industry standards in scalability. Through its optimized architecture and efficient use of operating systems, WebLOAD allows for unmatched levels of load generation.
- ◆ Low Virtual Client Footprint enables greater load generation on existing hardware infrastructure. Savings on infrastructure can fund test of higher loads.



4.1.10 Real-Time Performance Analysis for Better Insight

- ◆ Over 75 Performance Metrics to analyze and correlate
- ◆ System Metrics - WebLOAD can incorporate any data from Windows NT Performance Monitor or UNIX RSTATD directly into the test metrics for comprehensive analysis.
- ◆ Data Drilling - WebLOAD's unique Data Drilling feature enables you to view a global or detailed account of transaction successes and failures on an individual Virtual Client level. This feature is extremely useful in capturing intermittent errors.
- ◆ Regression Analysis - WebLOAD allows you to view a current running test vs. yesterday's test. WebLOAD allows you to graph any number of test results for accurate before/after comparison.

4.1.11 Database & Internet Protocol Testing

- ◆ Using a new interface, you can connect directly to SQL, Oracle and Access databases for query or load testing. Additionally, you can test non-HTTP protocols - FTP, UDP, TCP, SMTP, Telnet, POP, including SSL.

4.1.12 Security Testing

- ◆ Using a new engine in WebLOAD, you can launch Distributed Denial of Service attacks on your test application. Using WebLOAD for DDoS during the test cycle helps you ensure that your application can withstand malicious attacks once in production, and it gives you the opportunity to evaluate how your application will respond if subjected to such attacks. It includes common "attack" libraries that you can incorporate into your test agenda via an intuitive user interface.

4.1.13 System Requirements

Central Console: Microsoft Windows XP, 2000, 2003

CPU: Pentium III 800 and above

RAM: 512MB

Load Machine: Microsoft Windows XP, 2000, 2003, Sun Solaris 2.6 and above, Red Hat V7.3 and above

CPU: Pentium III 800 and above

RAM: 512MB

Probing Client: Microsoft Windows XP, 2000, 2003, Sun Solaris 2.6 and above, Red Hat V7.3 and above

CPU: Pentium III 500 and above

RAM: 512MB