# Table of Contents

1. Introduction ................................................................................................................. 3
2. Tools Used .................................................................................................................. 3
   2.1 AppMeter Feature Details ................................................................................... 3
3. General Summary ....................................................................................................... 4
   3.1 Types of Performance Testing ............................................................................ 4
   3.2 Reporting Metrics ............................................................................................... 5
   3.3 Keys to Accuracy ................................................................................................ 5
   3.4 Load Scalability .................................................................................................. 6
4. Process Steps............................................................................................................... 6
   4.1 Conducting a System Analysis ........................................................................... 6
   4.2 Usage Patterns – Understanding the Nature of the Load .................................... 6
   4.3 Usage Patterns – Determine User Session Variables / Page Request Distribution 7
   4.4 Determine Range & Distribution of values for these variables ......................... 7
   4.5 Estimating Target Load Levels ........................................................................... 7
   4.6 Develop Base Test Scripts – Load Test Design .................................................. 8
   4.7 Determine Frequency of Execution .................................................................... 8
   4.8 Execute Test Scripts ............................................................................................ 8
   4.9 Monitor Execution ............................................................................................... 9
   4.10 Collect & Examine Reports .............................................................................. 9
   4.11 Provide Reports and Suggestions ...................................................................... 11
1 **Introduction**

AppLabs Technologies provides offshore and on-site quality assurance services. We have a 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, AppLabs 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, including web-based applications.

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 can’t be found with conventional methods of testing.

2 **Tools Used**

A variety of tools are available for generating performance tests on web-based and client-server applications. Mercury Interactive’s LoadRunner® or Radview’s Webload® is recommended in most scenarios; however, some applications require the use of custom developed tools. AppLabs has developed AppMeter, a Load Testing tool based on the open source Java tool, JMeter. AppLabs can also develop a custom tool solution known as a “Test Harness” if it is required by the application under test.

2.1 **AppMeter Feature Details**

- **100% pure Java**
  - It is a 100% pure Java desktop application designed to load test functional behavior and measure performance.
  - Complete portability across different platforms.
  - Fully Swing based application and thus lightweight component support.

- **Multithreaded Framework**: Full multithreading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.

- **Unlimited virtual users**: Other tools charge on a per-virtual-user basis. With AppMeter, we can use an unlimited number of virtual users to create any size load that is needed without increasing costs or overhead.

- **Saving results into CSV file**: As the test results are saved into a CSV file (comma delimited/separated values), the statistical analysis of the test results is quickly and easily done with Excel or other CSV compatible software.

- **Statistical friendly test results**: The tool provides all the parameters required for statistical analysis. So finding the statistical significance of the test results is effortless.
- **Built-in log analyzers**: AppMeter comes with built-in tools such as an exception analyzer and word analyzers, which are instrumental in analyzing the log files of different servers.

- **Conditional Branching Support**: Not every load test will include a predetermined sequence of events. When the user-actions are based on previous actions (or correct, different results from the same actions), conditional branching is an absolute necessity. AppMeter supports conditional branching and allows the flexibility to test today’s dynamic applications.

- **Regular Expressions**: Using regular expressions, AppMeter can quickly and easily be used to find patterns, errors, and other text-intensive operations. Regular expressions are very handy in determining if a request fails based on the response. Another advantage of this feature is the ease of capturing the required values from a response; This can allow AppMeter to efficiently use these values for conditional navigation/branching or for using as an input for the next request.

- **HTTP Head Request Support**: This feature just checks for the availability of a particular resource (say images) instead of getting its content. In situations where we have to check for the availability of a resource, but not to download its content, this feature is superb.

- **SSL Support**: The tool supports HTTPS protocol as well as supporting many other protocols. It supports HTTP, FTP, JDBC, and SOAP etc.

- **Variety of Timers**: The most common timers used in any load testing are constant timer, uniform random timer and Gaussian random timer. The tool has all these built-in timers.

- **Many other Features**: Some of the other features available are counters, assertions, cookie manager, header manager etc.

### 3 General Summary

#### 3.1 Types of Performance Testing

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

- **Stress**: Load testing over an extended period of time. The objective is to validate an application’s stability and reliability.

- **Capacity**: 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.
3.2 Reporting Metrics

**Load size:** The number of concurrent Virtual Clients trying to access the site.

**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.

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

**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.

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

**Send Time:** The time it takes the Virtual Clients to write a request to the ABT (Application being tested), in seconds.

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

**Process Time:** The time it takes to parse a response from the ABT (Application being tested) and then populate the document-object model (the DOM), in seconds.

**Wait Time (Average Latency):** The time it takes from when a request is sent until the first byte is received.

**Receive Time:** The elapsed time between receiving the first byte and the last byte.

3.3 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
3.4 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.

4 Process Steps

Diagram 1: Explained in Detail Below

4.1 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

4.2 Usage Patterns – Understanding the Nature of the Load

Understanding where and when the load comes from is necessary to correctly simulate the load. With this information, minimum, maximum, and average loads
can be determined, as well as the distribution of the load. 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

4.3 Usage Patterns – Determine User Session Variables / Page Request Distribution

What processes that the load triggers is needed to simulate the correct load. Some 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
  Examples:
  - “Home page”
  - Product info page
  - Credit Card Info page
- Varying User Activities from page types.
  Examples:
  - Uploading information
  - Downloading information
  - Purchasing
- Page Rendering Issues
  Examples:
  - Dynamic pages (such as time sensitive Auction/trade pages)
  - Static pages
- New users vs. Existing users

4.4 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 (e.g., 4pg views per session)
- Standard deviation
- Discrete distribution

4.5 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
4.6 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.

a. **Test Objective**
   SAMPLE: The objective of this load test is to determine if the Web site, as currently configured, will be able to handle the 12,000-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.

b. **Pass / Fail Criteria**
   SAMPLE: The load test will be considered a success if the Web site will handle the target load of 12,000 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.

c. **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
     SAMPLE:  
     1. Browse NoBuy: Home >> Product Information(1) >> Product
     2. Information(2) >> Exit
     3. View Calendar: Home >> Event Calendars Page >> Next 30 days >> back >> Exit
   - 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.

4.7 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.

4.8 Execute Test Scripts

Combine Scripts to create load-testing Scenario

   a. Scripts Executed
b. Percentages in which those scripts are run
c. Description of how load will be ramped up

**Sample Table 1: Script Table for static eCommerce site**

<table>
<thead>
<tr>
<th>Script Name</th>
<th>Basic Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Only</td>
<td>Home ➔ Exit</td>
</tr>
<tr>
<td>BrowseNoBuy</td>
<td>Home ➔ Product Information(1) ➔ Product Information(2) ➔ Exit</td>
</tr>
<tr>
<td>BrowseAndBuy</td>
<td>Home ➔ Product Information(1) ➔ Product Information(2) ➔ Order Entry ➔ Credit Card Validation ➔ Order Confirmation ➔ Exit</td>
</tr>
</tbody>
</table>

**Sample Table 2: Spreadsheet (script table vs. user frequency)**

<table>
<thead>
<tr>
<th>Script Name</th>
<th>Relative Frequency*</th>
<th>Home</th>
<th>Product Info</th>
<th>Order Entry</th>
<th>Credit Card</th>
<th>Order Confirmation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Only</td>
<td>55%</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BrowseNoBuy</td>
<td>34%</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BrowseAndBuy</td>
<td>11%</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Relative pg Views</th>
<th>Home</th>
<th>Product Info</th>
<th>Order Entry</th>
<th>Credit Card</th>
<th>Order Confirmation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>.9</td>
<td>.11</td>
<td>.11</td>
<td>.11</td>
<td>2.23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Resulting pg view distribution</th>
<th>45%</th>
<th>40%</th>
<th>5%</th>
<th>5%</th>
<th>5%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Target pg view distribution</td>
<td>48%</td>
<td>37%</td>
<td>6%</td>
<td>5%</td>
<td>4%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Difference**</td>
<td>3%</td>
<td>-3%</td>
<td>1%</td>
<td>0%</td>
<td>1%</td>
<td></td>
</tr>
</tbody>
</table>

* Obtained from historic log files or created from projected user patterns
** Target Difference should be < 5% for non-transaction pages and <2% for transaction pages.

4.9 **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.

4.10 **Collect & Examine Reports**

Graphs: AppLabs utilized a suite of graphs from either LoadRunner, WebLoad, AppMeter, or other customized performance testing tool to provide the client with easy to use, relevant information.
Sample Graph 1: Transaction Summary

This performance graph displays the number of transactions that passed, failed, aborted or ended with errors. For example, these results show the Submit_Search business process to have passed all its transactions at a rate of approximately 96%.

Sample Graph 2: Throughput

This web graph displays the amount of throughput (in bytes) on the Web server during load testing. This graph helps testers evaluate the amount of load X users generate in terms of server throughput. For example, this graph reveals a total throughput of over 7 million bytes per second.

Additional Report Info Available during & after Scenario Execution

The below are additional reporting metrics available to those item mentioned in ‘definitions’ section.

a. **Rendezvous** – Indicates when and how virtual users are released at each point.

b. **Transactions/sec (passed)** – The number of completed, successful transactions performed per second. (see sample table 2)

c. **Transactions/sec (failed)** – The number of incomplete, failed transactions per second. (see sample table 2)

d. **Percentile** – Analyzes percentage of transactions that were performed within a given time range. (see sample table 2)

e. **Performance Under Load** – Transaction times relative to the number of virtual users running at any given point during the scenario.
f. **Transaction Performance** – Average time taken to perform transactions during each second of scenario.

g. **Transaction Performance Summary** – Minimum, maximum, and average performance times for all transactions in scenario.

h. **Transaction Performance by Virtual User** – Time taken by an individual virtual user to perform transactions during the scenario.

i. **Transaction Distribution** – The distribution of the time taken to perform a transaction.

j. **Connections per Second** – Shows the number of connections made to the Web server by virtual users during each second of the scenario run.

4.11 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, AppLabs will recommend courses of action to resolve bottlenecks, increase scalability, and performance.