

**Step 1 - Identify Performance Acceptance Criteria**

Identifying performance acceptance criteria is most valuable when initiated early in the application's development life cycle. It is frequently valuable to record the acceptance criteria for your application and store them in a place and format that is available to the entire team for review and comment. Criteria are typically determined by balancing your business, industry, technology, competitive, and user requirements.

Test objectives frequently include the following:

· Response time.

For example, the product catalog must be displayed in less than 3 seconds.

· Throughput.

For example, the system must support 100 transactions per second.

· Resource utilization.

A frequently overlooked aspect is the amount of resources your application is consuming, in terms of processor, memory, disk input output (I/O), and network I/O.

. Maximum user load.

This test objective determines how many users can run on a specific hardware configuration.

· Business related metrics. This objective is mapped to business volume at normal and peak values; for example, the number of orders or Help desk calls handled at a given time.

**Step 2 - Identify Key Scenarios**

Scenarios are anticipated user paths that generally incorporate multiple application activities. Key scenarios are those for which you have specific performance goals, those considered to be high-risk, those that are most commonly used, or those with a significant performance impact. The basic steps for identifying key scenarios are.

**1. Identify all the scenarios for a Web application.**

For example, even the most basic e-commerce application must support the following user scenarios:

o Browse catalog

o Search for a product

o Place an order

**2. Identify the activities involved in each of the scenarios.**

For example, a "Place an Order" scenario will include the following activities:

o Log on to the application.

o Browse the product catalog.

o Search for a specific product.

o Add items to the shopping cart.

o Validate credit card details and place an order.

**3. Identify the scenarios that are most commonly executed or most resource-intensive; these will be the key scenarios used for load testing.**

For example, in an e-commerce application, browsing a catalog may be the most commonly executed scenario, whereas placing an order may be the most resource-intensive scenario because it accesses the database.

o The most commonly executed scenarios for an existing Web application can be determined by examining the log files.

o The most commonly executed scenarios for a new Web application can be obtained from market research, historical data, market trends, and so on.

o Resource-intensive scenarios can be identified by using design documents or the actual code implementation. The primary resources are:

Processor

Memory

Disk

I/O

Network I/O

Once they have been identified, you will use these key scenarios to create workload profiles and to design load tests.

**Step 3 - Create a Workload Model**

When defining workload distribution, consider the following key points for determining the characteristics for user scenarios:

· A user scenario is defined as a navigational path, including intermediate steps or activities, taken by the user to complete a task. This can also be thought of as a user session.

· A user will typically pause between pages during a session. This is known as user delay or think time.

· A session will have an average duration when viewed across multiple users. It is important to account for this when defining the load levels that will translate into concurrent usage, overlapping users, or user sessions per unit of time.

· Not all scenarios can be performed by a new user, a returning user, or either; know who you expect your primary users to be and test accordingly.

**Step 4 - Identify Target Load Levels**

Identify the load levels to be applied to the workload distribution(s) identified during the previous step. The purpose of identifying target load levels is to ensure that your tests can be used to predict or compare a variety of production load conditions.

The following are common inputs used for determining target load levels:

· Business volume (both current and projected) as it relates to your performance objectives

· Key scenarios

· Distribution of work

· Session characteristics (navigational path, duration, percentage of new users)

By combining the items above, you can determine the remaining details necessary to implement the workload model under a particular target load.

**Step 5 - Identify Metrics**

There is a virtually unlimited number of metrics that can be collected during a performance test execution. However, collecting too many metrics can make analysis unwieldy as well as negatively impact the application's actual performance. For these reasons, it is important to identify the metrics that are most relevant to your performance objectives and those that you anticipate will help you to identify bottlenecks. Only well-selected metrics that are analyzed correctly and contextually provide information of value.

The following are a few suggestions for identifying the metrics that will provide the most valuable information to your project:

· Define questions related to your application performance that can be easily tested.

For example, what is the checkout response time when placing an order?

How many orders are placed in a minute? These questions have definite answers. · With the answers to these questions, determine quality goals for comparison

against external expectations. For example, checkout response time should be 30 seconds, and a maximum of 10 orders should be placed in a minute. The answers are   
based on market research, historical data, market trends, and so on.

·

**Identify the metrics.** Using your list of performance-related questions and answers,   
identify the metrics that provide information related to those questions and answers.

·

**Identify supporting metrics.** Using the same approach, you can identify lower-level   
metrics that focus on measuring the performance and identifying the bottlenecks in   
the system. When identifying low-level metrics, most teams find it valuable to   
determine a baseline for those metrics under single-user and/or normal load   
conditions. This helps you determine the acceptable load levels for your application.   
Baseline values help you analyze your application performance at varying load levels   
and serve as a starting point for trend analysis across builds or releases.

·

**Reevaluate the metrics to be collected regularly.** Goals, priorities, risks, and   
current issues are bound to change over the course of a project. With each of these   
changes, different metrics may provide more value than the ones that have previously   
been identified.

Additionally, to evaluate the performance of your application in more detail and to   
identify potential bottlenecks, it is frequently useful to monitor metrics in the following   
categories:   
·

**Network-specific metrics.** This set of metrics provides information about the overall   
health and efficiency of your network, including routers, switches, and gateways.

·

**System-related metrics.** This set of metrics helps you identify the resource   
utilization on your server. The resources being utilized are processor, memory, disk   
I/O, and network I/O.

·

**Platform-specific metrics.** Platform-specific metrics are related to software that is   
used to host your application, such as the Microsoft .NET Framework common   
language runtime (CLR) and ASP.NET-related metrics.

·

**Application-specific metrics.** These include custom performance counters inserted   
in your application code to monitor application health and identify performance   
issues. You might use custom counters to determine the number of concurrent threads   
waiting to acquire a particular lock, or the number of requests queued to make an   
outbound call to a Web service.

·

**Service-level metrics.** These metrics can help to measure overall application   
throughput and latency, or they might be tied to specific business scenarios.

·

**Business metrics.** These metrics are indicators of business-related information, such   
as the number of orders placed in a given timeframe.

**Step 6 - Design Specific Tests**

Using your scenarios, key metrics, and workload analysis, you can now design specific   
tests to be conducted. Each test will generally have a different purpose, collect different   
data, include different scenarios, and have different target load levels. The key is to   
design tests that will help the team collect the information it needs in order to understand,   
evaluate, or tune the application.   
  
Points to consider when designing tests include:

Do not change your test design because the design is difficult to implement in your   
tool.

·

If you cannot implement your test as designed, ensure that you record the details   
pertaining to the test that you do implement.

·

Ensure that the model contains all of the supplementary data needed to create the   
actual test.

·

Consider including invalid data in your performance tests. For example, include some   
users who mistype their password on the first attempt but get it correct on a second   
try.

·

First-time users usually spend significantly more time on each page or activity than   
experienced users.

·

The best possible test data is test data collected from a production database or log file.

·

Think about nonhuman system users and batch processes as well as end users. For   
example, there might be a batch process that runs to update the status of orders while   
users are performing activities on the site. In this situation, you would need to account   
for those processes because they might be consuming resources.

·

Do not get overly caught up in striving for perfection, and do not fall into the trap of   
oversimplification. In general, it is a good idea to start executing tests when you have   
a reasonable test designed and then enhance the design incrementally while collecting   
results.

**Step 7 - Run Tests**

Poor load simulations can render all of the work in the previous activities useless. To   
understand the data collected from a test execution, the load simulation must reflect the   
test design. When the simulation does not reflect the test design, the results are prone to   
misinterpretation. Consider the following steps when preparing to simulate load:   
  
1.

Configure the test environment in such a way that it mirrors your production   
environment as closely as possible, noting and accounting for all differences between   
the two.

2.

Ensure that performance counters relevant for identified metrics and resource   
utilization are being measured and are not interfering with the accuracy of the   
simulation.

3.

Use appropriate load-generation tools to create a load with the characteristics   
specified in your test design.

4.

Using the load-generation tool(s), execute tests by first building up to the target load   
specified in your test design, in order to validate the correctness of the simulation.   
Some things to consider during test execution include:

o

Begin load testing with a small number of users distributed against your user   
profile, and then incrementally increase the load. It is important to allow time   
for the system to stabilize between increases in load while evaluating the   
correctness of the simulation.

o

Consider continuing to increase the load and record the behavior until you   
reach the threshold for the resources identified in your performance   
objectives, even if that load is beyond the target load specified in the test

design. Information about when the system crosses identified thresholds is just   
as important as the value of the metrics at the target load of the test.

o

Similarly, it is frequently valuable to continue to increase the number of users   
until you run up against the service-level limits beyond which you would be   
violating your SLAs for throughput, response time, and resource utilization.

**Note:**Make sure that the client computers (agents) you use to generate load are not   
overly stressed. Resource utilization such as processor and memory must remain well   
below the utilization threshold values to ensure accurate test results.

**Step 8 - Analyze the Results**

You can analyze the test results to find performance bottlenecks between each test run or   
after all testing has been completed. Analyzing the results correctly requires training and   
experience with graphing correlated response time and system data.   
  
The following are the steps for analyzing the data:   
  
1.

Analyze the captured data and compare the results against the metric's accepted level   
to determine whether the performance of the application being tested shows a trend   
toward or away from the performance objectives.

2.

Analyze the measured metrics to diagnose potential bottlenecks. Based on the   
analysis, if required, capture additional metrics in subsequent test cycles. For   
example, suppose that during the first iteration of load tests, the process shows a   
marked increase in memory consumption, indicating a possible memory leak. In the   
subsequent iterations, additional memory counters related to generations can be   
captured to study the memory allocation pattern for the application.

**Summary**

Load testing helps to identify the maximum operating capacity of the application and any   
bottlenecks that might be degrading performance.   
  
The basic methodology for performing load testing on a Web application is to identify the   
performance-critical key scenarios; identify the workload profile for distributing all the   
load among the key scenarios; identify metrics that you want to collect in order to verify   
them against your performance objectives; create test cases that will be used to simulate   
the load test; use tools to simulate the load according to the test cases and capture the   
metrics; and finally, analyze the metrics data captured during the tests.

**Workload Modeling and Profiles for Load Testing**

## What is the set of possible actions that a user can perform?

This depends on the specifics of your web-application. For a typical e-commerce application, consider the following actions:

* Connect to the home page.
* Log on to the application.
* Browse the product catalog.
* Search for specific products.
* Add products to the shopping cart.
* Validate and place an order.
* Log out from the application.

## What are user profiles for the application? Will a single user profile be enough?

 if a load test employs multiple user profiles, it is best to debug each of the profiles separately. Once you've confirmed that each profile performs adequately on its own, you can begin combining multiple profiles into a single test.

## What are the actions performed by a user representative of each profile?

|  |  |
| --- | --- |
| **ser Scenario** | **Percent Load Distribution** |
| Browsing product catalog | 40 |
| Creating a user account | 05 |
| Searching for a product | 30 |
| Login to application | 15 |
| Order Placement | 10 |
| **Total** | **100** |

## What is the average user “think time” between actions?

The time spent by the user between two consecutive actions on a page is called “think time” in load testing parlance. During “think time,” the user may be viewing the information displayed on a page or may be entering details such as addresses, credit card numbers, search parameters, etc.

Think time can vary depending on the complexity of the information on a page. For example, the think time for a logon page is less than the think time for an order placement page where a user must enter credit card details.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **enario** | **Action** | **Input Data** | **Output Data** | **Think Time** |
| Browsing Product Catalog by an existing user | Login | * Unique username * Password of the username |  | 5-8 seconds |
|  | Browse | * Catalog Tree * User Type | * Product description * Title * Category | 4-30 seconds |
| Browsing Product Catalog by a new user | Login | * Unique username * Password of the username |  | 5-15 seconds |
|  | Browse | * Catalog Tree * User Type | * Product description * Title * Category | 10-60 seco |

## What is the expected ratio of user profile scenarios?

## How does the number of users logged on to your site vary with time?

Ramp up and Ramp down

## What is the maximum expected number of users logged in to your application?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **me Period** | **Normal Load Requests (Avg.)** | **Peak Load Requests (Avg.)** | **Peak Load Build up Time** | **Peak Load Duration** |
| One Month | 72,000 | 2,16,000 | 2 Hours | 1 Hour |
| One Week | 16,800 | 50,400 | 2 Hours | 1 Hour |
| One day | 2,400 | 7,200 | 2 Hours | 1 Hour |
| One Hour | 100 | 300 | 2 Hours | 1 Hour |

## What is the duration for which the test needs to be executed?

Before designing workload model, it is important to collect relevant data which helps us to create effective workload model. Following items are required, in order to design effective workload model.

* Number of concurrent users
* Total Transactions to be achieved
* Scenario and its Actions
* % of total user for an action

Consider a web application which publishes your post across social medias; total number of concurrent during peak business hours (from 11:00 to 13:00) per day is 1500. Following are the list of transactions performed by the users.

1. Login
2. Click on Write Post link
3. Write Post (140 characters)
4. Submit Post
5. Logout

Our objective is to design workload model to perform load testing. From the available data, I have arrived workload model as shown below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Scenario** | **Actions** | **Transactions in seconds** | **Concurrent Users** | **Transactions to be achieved** | **% of total user** |
| 1 | **Submit Post Scenario** with 3 iterations with 900 seconds ramp-up period | Login | 5 | 1500 | 1500 X 3 = 4500 | 100 |
| 2 | Click on Write Post link | 3 |
| 3 | Write Post (140 characters) | 120 |
| 4 | Submit Post | 3 |
| 5 | Logout | 3 |

Above table has a scenario **Submit Post**with 3 iterations along with 15 minutes of ramp up period, hence the transaction to be achieved will be 4500. There is 100% load distribution across all the actions. Now next step is to prepare the graph for load testing.