HP LoadRunner

Load Testing Scenarios Best Practices

Table of contents

Welcome to Load Testing Scenarios Best Practices

This document provides concepts and guidelines for designing and executing load testing scenarios, including scenarios involving large numbers of virtual users, large amounts of data, and long running scenarios.

This document is intended for performance test automation engineers who are developing load testing scenarios.

Introduction to Load Testing Scenarios

Modern system architectures are complex. While they provide an unprecedented degree of power and flexibility, these systems are difficult to test. Whereas single-user testing focuses primarily on the functionality and the user interface of a system component, application testing focuses on the performance and reliability of an entire system.

For example, a typical application testing scenario might depict a thousand users that log in simultaneously to a system on Monday morning. What is the response time of the system? Does the system crash? To be able to answer these questions and more, a complete application performance testing solution must do the following:

- Test a system that combines a variety of software applications and hardware platforms
- Determine the suitability of a server for any given application
- Test the server before the necessary client software has been developed
- Emulate an environment where multiple clients interact with a single server application
- Test an application under the load of tens, hundreds, or even thousands of potential users

A load testing *scenario* defines the events that occur during a testing session. It defines and controls the number of users to emulate (known as *virtual users*, or *Vusers*), the actions that they perform, and the machines on which they run their emulations.

When the number of Vusers reaches the thousands, the resources required to run the scenario might come close to, or even exceed, the available computing resources. For example, the computers used to simulate the load (known as the *Load Generators*) might start to run out of memory, or there might not be enough CPU available to serve all of the Vusers.

When running a long load test, it can be very frustrating for the test to run for hours only to have something fail – especially if it's something other than the application under test. It is generally possible to mitigate this by planning the load test correctly. This document offers a number of strategies which can be adopted to help you achieve tests involving scenarios exceeding a thousand[1] Vusers.

General Recommendations

Here are some general recommendations that all load tests should follow:

- Set *TMP* and *TEMP* environment variables to paths without spaces
- Turn off all anti-virus and anti-spyware that are configured to scan whenever a file is accessed.
- Disable any scheduled scans that might take place during the test
- Ensure that no other unnecessary applications and processes are running
- Make sure there is plenty of free disk space on the Load Generator and Controller machines.
- Reboot all Load Generators prior to the test
- Don't run any Vusers on the Controller machine

[1] More than 5000 Vusers is generally considered a large test when using the Web protocol. For other protocols, large tests involve a thousand users or more.

Scripting

Every load test requires scripts which represent the end-user business processes that are executed by the Vuser during the load test. Here is a list of guidelines that you should follow when preparing Vuser scripts.

- All script code should be inside transaction delimiters
  - The transaction should also include suitable verification logic. Verification logic doesn't normally affect the response time, and will fail the transaction if the verification fails, so the transaction becomes more reliable.
- Add think time *before* the first transaction of each action. Think times simulate the natural delays caused by a user in between steps. This avoids tight looping in case of errors, such as when the application's home page cannot be reached. This can also be handled by using suitable pacing between iterations (this can be configured in the 'Pacing' branch of the Runtime settings dialog).
- Don't use the `sleep` function. Use `lr_think_time` instead, as it cedes control to LoadRunner, and allows LoadRunner the opportunity to continue working while the Vuser waits.
- Use hard-coded think time values. The actual think time that will be used can be adjusted in the runtime settings by using random percentages of recorded think time, so you won't need to update it in the script.
- Always use think times between transactions, unless the purpose is to severely stress test the application.
  - But ensure that there are no think times *inside* the transaction
- If your script contains `while` loops, increment and monitor a counter so that you can break out of the loop in case the loop's condition isn't reached.
- Remove all logging and output messages, such as the following:
  - `lr_vuser_status_message`
  - `lr_output_message`
  - `lr_error_message`
  - `lr_log_message`
  - `lr_message`

  If you can't remove them completely, keep them to a minimum. If you used them for debugging the script, remember to comment them out when you're done debugging.
- Prepare enough data in advance to cover the test's needs. This is particularly important when the script needs unique data or data that can only be used once.
  - You can use the Virtual Table Server (VTS), which is a web-based application that offers an alternative to standard LoadRunner parameterization. VTS allows you to assign parameter values from a single set of parameter values to multiple Vusers, whereas standard LoadRunner parameterization requires that each Vuser is assigned parameter values from a dedicated set of values.
- If you're using the Web protocol, and the pages accessed do not contain resources, you can reduce the load on the Load Generator by setting the *Mode* attribute to `"Mode=HTTP"` in `web_url` steps. This will avoid parsing the HTML, since the default mode is `"Mode=HTML"`.
- In situations where there may be many consecutive failures, you can detect these consecutive failures, and if more than some predetermined number of consecutive failures is encountered, call lr_abort to stop the script's execution.

Runtime Settings

Runtime settings define the way a Vuser script runs, and are applied to Vusers when you run a script using VuGen, the Controller, Performance Center, or Business Process Monitor. Note that the runtime settings which are available depend on the specific protocol that the Vuser script uses, so some of the recommendations below may not apply to all scripts.

- Disable all logging by ensuring that the *Enable logging* checkbox in the runtime settings dialog's *Log* branch is unchecked, unless you need it for debugging purposes (and with no more than 10 Vusers). Running many Vusers with logging can affect the test results because logging is a time-consuming activity. Each log message also takes up space on the disk, and in extreme circumstances, particularly on long tests, could cause the disk to fill up.
  - You should also disable the *Send messages only when an error occurs* option, since there may be a lot more errors than expected.
- Disable the *Generate snapshot on error* setting (in the *Error Handling* section of the runtime settings dialog's *Miscellaneous* branch) as well, since this also creates overhead on Load Generators.

- Run the Vusers as a thread (in the *Multithreading* section of the runtime settings dialog's *Miscellaneous* branch) to enable as many Vusers as possible to be run on each Load Generator, unless the protocol you're using requires them to be run as a process.
- Disable the *Define each step as a transaction* setting (in the *Automatic Transactions* section of the runtime settings dialog's *Miscellaneous* branch).
- Uncheck the *Simulate browser cache* setting (in the *Browser Emulation* section of the runtime settings dialog's *Browser* branch)
- Check the *Simulate new user on each iteration* in the runtime settings dialog's *Replay* branch.
- As mentioned in the Scripting section, avoid tight looping by using suitable pacing between iterations (in the Runtime settings dialog's '*Pacing*' branch).
- If you manually start Vusers (using the *Run/Stop Vusers* button or menu) they will run according to the *Number of Iterations* setting (in the *Iteration Count* section of the Runtime settings dialog's *Pacing* branch). If you start Vusers manually as part of a test involving a scheduled scenario, consider setting the value of *Number of Iterations* to a high number (eg. 9999) to ensure that the manually started Vusers continue to create load on the system while the scheduled Vusers are running.
- Never ignore think times. Think times simulate the delays caused by a real user 'thinking' in between steps. In a large load test, the aim is to simulate times as accurately as possible, and ignoring think time might cause the application under test to choke. Ensure that the *Replay think time* radio button in the *Think Time options* section of the runtime settings dialog's *Think Time* branch is selected, and configured appropriately.
- When using the Web protocol, disable Content Checks during replay wherever possible, especially if no rules are configured (in the Runtime setting dialog's '*Internet Protocol/ContentCheck*' branch.

Diagnostics and Monitoring Settings

Some diagnostics and monitoring settings can inadvertently increase the load on the Controller machine. Here are some guidelines to avoid this.

- Go to the Controller's *Diagnostics > Configuration* dialog to disable web page breakdown.
- Modify the monitors' sampling rate to reduce CPU utilization on the Controller. Go to *Tools > Options…*, and set the *Monitors* tab to the following:
  - Check the *Enable Transaction Monitor* checkbox in the *Transaction Data* section, and set *Frequency* to 10
  - Set the *Data Sampling Rate* in the *Server Resource Monitors* section to 10 sec.
  - Ensure the *Error Handling* option is set to *Send errors to the Output window*.
  - Ensure *Display debug messages* in the *Debug* section is unset.
- The output window (from the *View > Show Output* menu) should not be opened, as it causes unnecessary stress on the Controller.
- Monitor the following Key Performance Indicators (KPIs) on the Load Generator machines
  - Private bytes of the *mmdrv* process, to keep track of its memory consumption
  - Disk space
  - CPU usage
  - Network utilization

Scheduler

The scheduler allows you to accurately portray user behavior by scheduling Vusers to start and stop running within a certain time frame.

- Keep ramp-up speeds reasonable. Be careful if a large and increasing number of Vusers gets into the *pending* state rather than *Init*, *Ready* or *Running* during the ramp-up, since this could be an indicator of excessive ramp-up speeds. Reduce the ramp-up speed if necessary.
- When ramping up large numbers of Vusers very quickly, use the *Initialize each Vuser just before it runs* setting (in the *Edit Action* dialog box) in order not to overwhelm the connection between the Controller and the Load Generators. If starting all Vusers simultaneously, gradual initialization is recommended (by using the *Edit Action* dialog box from the Scheduler).
- When large scripts are used (for example, scripts with big parameter files), the initialization of the Vusers may take some time. If an aggressive ramp-up schedule is defined, the amount of Vusers in the *Init* state will build up, so that once the

scripts have been transferred and initialized, a large number will start running simultaneously. To avoid this, start a Vuser from each group manually on each Load Generator before the ramp-up starts.

- Avoid adding a high number of Vusers per ramp-up step. Instead, ramp up in intervals between five and fifteen seconds. For example, in order to ramp up 500 Vusers per minute for 30 minutes, ramp up 50 Vusers every six seconds rather than 500 per minute.
  - You can also initialize Vusers before ramp-up.

Analysis

Analyzing the results is an essential part of the load testing process. Here are some points to note.

- Always collate the results of the load test so they can be analyzed.
- Configure Analysis to use SQL Server as the database (from the *Database* tab of the *Options* dialog) to ensure faster analysis and higher data accuracy
- When analyzing results of large load tests:
  - Analyze complete data by default to avoid waiting for the summary information, and then waiting again for complete data (from the *Data source* section of the *Result Collection* tab of the *Options* dialog)
  - Analyze the results using an empty Analysis template. This will reduce the time to create the graphs saved in a non-empty template.
  - When opening a graph, filter it first to reduce the time it takes to open the graph.

Load Tests in HP Performance Center

HP Performance Center is a Web-based load test management framework that helps you centralize resources and collaborate across distributed teams and projects. It also offers the ability to schedule time on multiple LoadRunner resources for test execution. Here are some guidelines to help you achieve your goals in Performance Center.

- Timeslots are Performance Center's mechanism for reserving computing resources (Load Generator machines, Controller machines, etc.), for conducting load testing at specific scheduled times. These resources might be shared with other projects, so if you reserve timeslots in advance, make sure you free them up when you're finished testing
  - If you need to cancel a test, remember to cancel the timeslot as well in order to free up the resources.
- You should reserve a timeslot for each test. However, reserving a single timeslot spanning multiple days to run multiple small tests can cause host resource availability issues for other users and should be avoided.
- Reserve a timeslot window long enough to account for the ramp-up of the large number of Vusers.
- Avoid reserving more hosts than the test will use. This will result in unused host resources being reserved, and will prevent other users from running load tests due to lack of testing resources.
- Enable the *Pause the Scheduler at load test start* checkbox to make sure that all teams participating in the load test are ready before the actual ramp-up of the Vusers begins.
  - Using *Pause the Scheduler at load test start* will also prevent any fast ramp-ups from starting before the run page UI is available.
- When the test is about to run, check whether the reserved hosts are actually available and in an operational state.
- After the test is finished, make sure to free up the timeslot and analyze any results that haven't yet been collated.

Application Under Test

We have considered many aspects of the load test, but we should also pay attention to the application whose load we need to test. Here are some simple steps that you can take to help ensure the success of the load test.

- Ensure that the application under test (AUT) is up and running on a production, or production-like environment. The most accurate results will be obtained if the load test is performed on an environment that is as close as possible to the actual production environment. Check that the operating systems, databases and web servers on all of the machines that make up the AUT have the correct versions installed, along with all necessary updates and patches.
- Ensure that you have installed the correct version of the AUT, and that all necessary patches and updates have been applied.

- If the complete AUT environment cannot be installed, for example, if it depends on third-party services, or production services that cannot be stressed for testing purposes, use Service Virtualization to simulate the missing service, and configure it to use a performance model that mimics real-life performance
- Make sure that the database statistics for all databases used by the AUT are up-to-date before the test is run
- Configure all logging in the AUT to only log errors.
- Ensure that the only users that will be using the AUT during the load test are Vusers that are scheduled as part of the load test. Casual users that log on during the test can skew the results and make the load test inaccurate.

Planning Load Tests

Here are some guidelines which you should follow whenever you plan a load test.

- Know your numbers: You should establish target metrics for your load test in advance. Knowing your goals and the numbers you want to reach will enable you to set baseline criteria that you can measure against in your tests. Some examples of metrics include Hits per Second, Transactions per Second, Transaction Response Time, etc. However, performance engineers will sometimes run large load tests while intentionally ignoring metrics until the system fails.
- Virtual User Profiling: Typically, you will have business requirements for the amount of load that your application is expected to handle. You should determine the number of virtual users per script that should be exercising the business process in order to achieve the required load. You can then calculate how to distribute the Vusers across all the scripts both by number and percentage. This will make it easy to create your scenario as all the information you will need is there.
- Data Preparation: You should figure out the amount of data needed to drive the load for at least an hour beyond the expected length of the session. This will ensure that you don't run out of data if some kind of delay occurs. Once the data is generated and the database is seeded with the conditions to support the load test, back up your database. This will allow you to quickly restore the data after the load test and make the environment ready to use again.
  - You can also use the Virtual Table Server (VTS), described in the Scripting section, to manage the data and how it is used in the load test.
- Load Testing Participants: Make sure the right experts and stakeholders are available when the load test is being executed, so that they can see how their area of responsibility behaves under actual load. These experts are typically network, database, and Web/application administrators. Typically application developers are not present, but should be 'on call' in case they are needed.

Planning Large Scenarios

Issues are sometimes encountered when the load test produces a large number of errors. When there are more than a thousand errors per second, the Controller can become overloaded and behave unpredictably. Similar effects can be observed when large amounts of online data are produced.

These issues can be mitigated by splitting the scenario into a number of different groups, where one group behaves as normal, dealing with data and writing messages, while the other group has its data and messages disabled. For example, let's assume our scenario consists of a single group of a thousand Vusers. We can split this into two groups:

- *Group1* – 100 Vusers
- *Group2* – 900 Vusers

Together, these groups will create the same load as the original scenario. But we can configure *Group2* to ignore data sending and writing, and to avoid sending error messages and log messages to the Controller. This is done by selecting the group in the Controller's *Design* tab, right-clicking, and selecting *Details…* In the dialog box that opens, click on the *More* button to reveal the *Command line* edit box. Add the following to the edit box:

```
-disable_data -disable_messages
```

Note that each of these instructions is preceded by a minus sign. They behave as follows:

- **`-disable_data`**: Instructs the group not to send any online data, and not to write any offline data
- **`-disable_messages`**: Instructs the group not to send any messages (errors, logs) to the Controller.

The consequences of using these command-line options are that no offline or online data will be sent to the Controller for that group. This means that no analysis data will be available for the Vusers in that group.

You should consider reserving a separate Load Generator for running a single Vuser which you can use as a control Vuser. You can then use the data reported by the control Vuser as a reference to ensure that the data reported by the other Vusers is accurate.

Determining the Maximum Number of Vusers per Load Generator

It is not possible to provide an absolute number of Vusers per Load Generator, as each Vuser script has its own unique behavior and requirements. But you can perform some calculations to get a value for your specific situation, as follows.

First, identify a single Vuser footprint:

- Create a scenario in the Controller which uses your script, and set the number of iterations to 30 in the Runtime Settings dialog. It should be configured to run until complete.
- Using the *Windows Resource Monitor*, configure *%Process Time* and *Private Bytes* counters on the *mdrv.exe* process in order to measure the footprint. Note that in order to configure these counters, you should first run the script to have *mdrv.exe* running. Run the scenario for the sake of configuring the counters and then stop and save it before initiating the real test.
- Run the scenario and make sure the monitor data is collected
- As soon as the scenario execution ends, analyze the results using the Analysis tool. Make sure you analyze complete data and not only summary data.
- Open the Windows Resource graph and make a note of the average CPU and peak memory utilization.

Now you can determine the number of Vusers per Load Generator. You need to work out how many Vusers the CPU can sustain, and how many the memory can accommodate.

- This formula will give you the number of Vusers the CPU can sustain (limited to 70% so as not to overuse the CPU):

$$\boldsymbol{Number\ of\ Vusers\ per\ CPU = \frac{70\% * Number\ of\ Core\ Processors}{Vuser\ Average\ CPU\ Usage}}$$

- This formula gives you the number of Vusers the memory can accommodate:

$$\textbf{Number of Vusers per Memory} = \frac{\textbf{Total GB RAM of Load Generator} - \textbf{GB RAM for the OS and other processes}}{\textbf{Vuser Peak Memory Usage}}$$

The number of Vusers that you need is the *lower* of these two values.

Here's a table with some examples, where 1GB is reserved for the Operating System and other processes:

No of core processors	Average CPU usage of Vuser (%)	Vusers for CPU (70%*No. of core processors)/Vuser Average CPU usage	Total assignable memory (GB) (Total memory – 1GB)	Peak memory per Vuser (MB)	Vusers for memory (Total assignable memory/Peak memory per Vuser)	Lower of *Vusers per CPU* and *Vusers by Memory*
8	10	(70*8)/10=56	8-1=7	80	7GB/80MB=89	56
4	22	(70*4)/22=12	8-1=7	100	7GB/100MB=70	12
8	6	(70*8)/6=93	16-1=15	120	15GB/120MB=125	93

Checklist

Here is a simple checklist that you can use as a basis for your load tests.

Area	Item	Status
Application Under Test (AUT)	Prepare environment with correct OS, Databases and Web Servers, including patches and updates	
	Install correct version of AUT	
	Install all patches and updates	
	Simulate unavailable services with appropriate performance model	
	Lock out users not participating in the load test	
	Configure logging of errors only	
	Expose performance metrics	
Databases (AUT, and those used by the load testing tools)	Rebuild all indexes	
	Collect statistics	
	Ensure there is enough disk space	
	Ensure the schema has enough space	
	Seed the AUT database with test data	
Controller	Ensure there is enough disk space	
	Configure Diagnostics and Monitoring Settings	
	Configure the Scheduler	
	Turn off all unnecessary services	
	Turn off all anti-virus and anti-spyware applications	
	Ensure that all Load Generators are connected	
Load Generator	Turn off all unnecessary services	
	Ensure there is enough disk space	
	Reboot all Load Generators prior to the test	
	Monitor the mmdrv process to keep track of its memory, CPU usage, disk space and network	
	Configure HPE Network Virtualization to ensure accurate simulation of real network conditions	
Performance Center	Reserve a timeslot which is long enough for the load test	
	Free up the timeslot after the test runs	

Summary

Load testing involves simulating demand on a system and measuring its response. Testing situations of particularly high demand over an extended period of time will not only stress the application under test, but also runs the risk of stressing the systems that are used to support the testing, such as Load Generators and Controllers. Failure of these systems prior to the test's completion can be very costly. Following the guidelines in this document will help minimize the risk of the test failing.