

Open Source Load Generation Tools: Test Scripting Challenges and their Mitigation Strategies

Abstract

Effective performance test scripting is a critical factor that dictates the success of any performance testing engagement. Performance test scripting is a complex process and unless one is equipped with the right tools and processes it is easy to go wrong at every turn. There are a wide range of load generation tools available in the market, both commercial and open source. While most open source tools effectively operate as “load generating engines”, they miserably lag behind commercial tools with respect to features like scenario recording, script debugging, customizable reports etc. Features related to scripting, such as recording, syntax highlighting IDE and visual debugging; often tip the scales in favor of these steeply priced commercial products.

It would be foolish to argue about the superior quality features provided by the commercial tools. But sometimes due to stringent budget we are forced to search for more pocket friendly solutions. There's no point wrinkling your nose on open source tools just because they are free. They have many advantages, for one, the technical support offered by the open source community is nothing to be laughed at. No doubt, there are a number of unique issues and challenges associated with doing Performance testing using Open Source Load generation tools, but they are not insurmountable; all that is required is a bit of innovative thinking and the right toolset.

In this paper, we examine the challenges faced during the performance test scripting phase of web applications using open source load generation tools. Additionally, we illustrate how the creative use of home grown tools and a methodical approach makes the test scripting phase efficient and effective.

Audience: Performance Test Managers and Leads,

Area of Application: Performance Testing of N-Tiered web based application

Benefits:

- 1.Improvement in performance test scripting efficiency while using cost effective open source load generation tools
- 2.Effective reduction in scripting and debugging time

Issues and Challenges:

Challenges in performance test scripting using open source load generation tools

Introduction

Early in the evolutionary phase of MindTree's [] Performance Testing Expert Service, we recognized the need to cater to the performance needs of an assorted customer base. It was evident that everyone did not have the deep pockets necessary to pay for exorbitantly priced commercial load generation tools. The necessity to provide a cost optimized solution was typically required in case of:

- a short performance testing cycle where the duration of testing doesn't justify large investment in a load generation tool
- budget constrained customer

Consciously pursuing a **tool independent approach** for testing web applications from the beginning enabled us to cope with diverse customer requirements and various engagement models. Instead of taking a rigid approach of imposing the same tool(s) across various engagements, we built parallel capabilities in commercial as well as open source solutions.

Most high end commercial tools provide a one stop solution for end to end performance testing. It would be foolhardy to state that open source tools can match the “quality of output” provided by commercial tools, out of the box. But having said this, it wouldn't be completely untrue to state that comparable results can be achieved at a fraction of the cost, using open source tools, provided you understand its strengths and weaknesses. The secret of successfully utilizing an

open source load generation tool is to develop an arsenal of helper kits, frameworks and processes to mitigate the identified shortcomings.

While most open source tools effectively operate as “load generating engines”, they miserably lag behind commercial tool offerings with respect to vital features like

- Scenario Recording/Script Debugging [Scripting Phase]
- Scaling up issues in distributed mode[] [Execution Phase]
- Lack of system monitoring [Execution Phase]
- Customizable report generation etc. [Reporting Phase]

Out the aforementioned issues, recording and scripting efficiency related problems with open source tools have been among the prime reasons of frustrations for performance test engineers.

This paper focuses on the challenges faced during the scripting phase (recording, scripting, debugging) for a web application performance testing engagement using open source tools like JMeter and OpenSTA and suggests strategies to overcome them.

Motivation

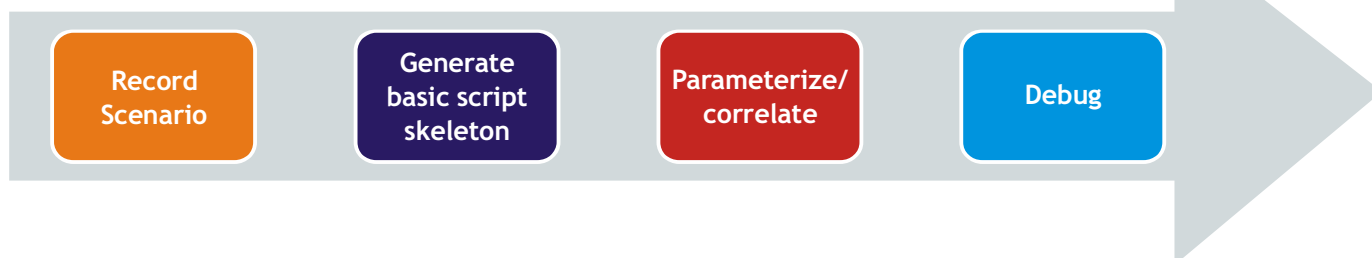
During the planning phase of numerous performance testing projects, we frequently had to evaluate open source tool based solution against commercial ones that could be effectively used across web applications. Most often the decisive questions took the form of:

- Does the open source tool have built-in recording features? Does it support the HTTPS protocol?

Traditional Performance Test Scripting Cycle

Before delving further into the scripting challenges, let's take a quick glance at a typical performance test scripting cycle.

The scripting phase heavily depends on the tool selection process. Typically, the tool selection activity is carried out during the strategy & planning phase. Once the tool is identified the activity of scripting load generation scenarios starts. The steps involved in the creation of load generation scripts are depicted below:



- How complex/complicated is the scenario workflow? Will the rudimentary scripting / debugging feature of the open source tool become the bottleneck?

Not surprisingly, the answers to these questions frequently tipped the balance in favor of commercial tools, even if the cost is outrageously high.

As part of the open source performance testing capability building strategy, we as a Performance Engineering team, have built comprehensive frameworks and helper tools around open source tools (especially JMeter, a commonly used open source load generation tool) to overcome their deficiencies. Interestingly, even we overlooked the scripting related challenges initially and focused more on aspects like improving monitoring, reporting and analysis features of open source load generation tools using helper/wrapper frameworks. We met our nemesis on a major Performance Testing engagement when a JMeter based open source solution went awry. We had underestimated the complexity of the workflow of scripts and each scenario took around 5-7 days to script instead of the typical 1.5 to 2 days. The problem was severely aggravated by poor debugging capabilities of JMeter. For example, a couple of scripts failed to execute because of certain uncorrelated parameters. The trouble was that the script consisted of 55 requests and the top 7 requests had more than 450 request parameters (GET/POST) amongst them. And using JMeter to find the missing/uncorrelated parameter manually was akin to finding a needle in a haystack.

This awakened us to the criticality of tackling the inherent shortcomings in open source load generation tools related to scripting, recording and debugging.

Scenario recording is generally achieved using either the tool's built in recording proxy or some third party recorder. The basic skeleton script is nothing but the initial draft automatically generated by the recorder.

The next stage involves script correlation. Correlation is the process of systematically replacing the dynamic values with variables. The final script undergoes the process of comprehensive debugging to verify successful execution.

Challenges Associated With Traditional Scripting Approach While Using Open Source Tools

The challenges involved in scripting using the traditional approach becomes painfully evident in case of open source load generation tools. Some of the critical issues are listed below:

Recording:

Most open source load generation tools have rudimentary scenario/protocol recorders. For example, until recently JMeter lacked the ability to record HTTPS sessions. Because of this disadvantage performance test engineers had to resort to 3rd party recording tools (e.g. badboy to record and generate JMeter skeleton scripts)

Correlation:

This is where most commercial tools score hands down over open source load generation tools. Automatic correlation is still an elusive feature in the open source tool market. Correlation is a very tedious activity considering the fact that the script engineer needs to go through each and every http request to identify parameters with dynamic values and then parameterize it.

Debugging:

The lack of advanced debugging capabilities (stepping/breakpoints/watches etc.) is every test engineers' worst nightmare. Hence the performance test engineer has to solely rely on trial and error technique to debug the script.

Tool Independent Scripting Techniques

In an attempt to master the aforesaid challenges we dug deeper for the root cause of these problems.

While researching we realized that scripting and debugging processes were generally perceived to be

tightly integrated with the load generation tool. But it need not be so. Open source load generation tools are effective at generating load (cost per virtual user), why not use it just for this purpose Generating Load. By abstracting out the tasks related to recording, scripting and debugging from the load generation tool and transferring them to a set of external helper entities we could effectively circumvent the identified challenges. The concept of separating out the scripting module from a load generation tool is universally applicable across any load generation tool making it tool independent. Based on this concept we could envisage a set of generic utilities that would help us surmount the issues of recording, scripting and debugging (associated with open source load generation tools).

And thus, the seeds of MindTree Performance Test Script Help-kit were sown. MindTree Performance Test Script Help-kit is a suite of independent helper utilities that aids the performance test engineer during the performance test scripting phase. The following sections describe the design and features of the Script Help-kit in greater detail.

MindTree Performance Test Scripting Help-kit

Having defined the need for designing the scripting help-kit, the next challenge was to identify the architecture and the tools that will contribute to this help-kit. After brainstorming we came up with a series of questions that would help us define a robust architecture. Some major questions are listed below:

What are the constituents of a performance testing script?

How can I capture the information required to create a script?

Can I automatically generate a script which is specific to each PT tool?

How do I debug and resolve the problems in PT scripts?

Let's see the significance of each question:

What are the constituents of any performance testing script?

Basically, to create a PT script for any given business scenario one should have detailed information about the scenario. In performance testing terminology, a business scenario is a sequence of different URL's

(technically known as web request). A web request can be further elaborated as a collection of vital information such as type of request, header parameters, query string parameters, post parameters etc.

A PT script is just a replay of these web requests. This implied that, if we could store all relevant details from the web request to some format, we could easily reuse it to create a PT script. This germinated the idea of defining a comprehensive storage format to store scenario related information which can be processed further by some tool (possibly home-grown) to create different scripts. We decided to use light-weight xml as the storage entity and baptized it as IntermediateFormat.

How can I capture the information required to create a script?

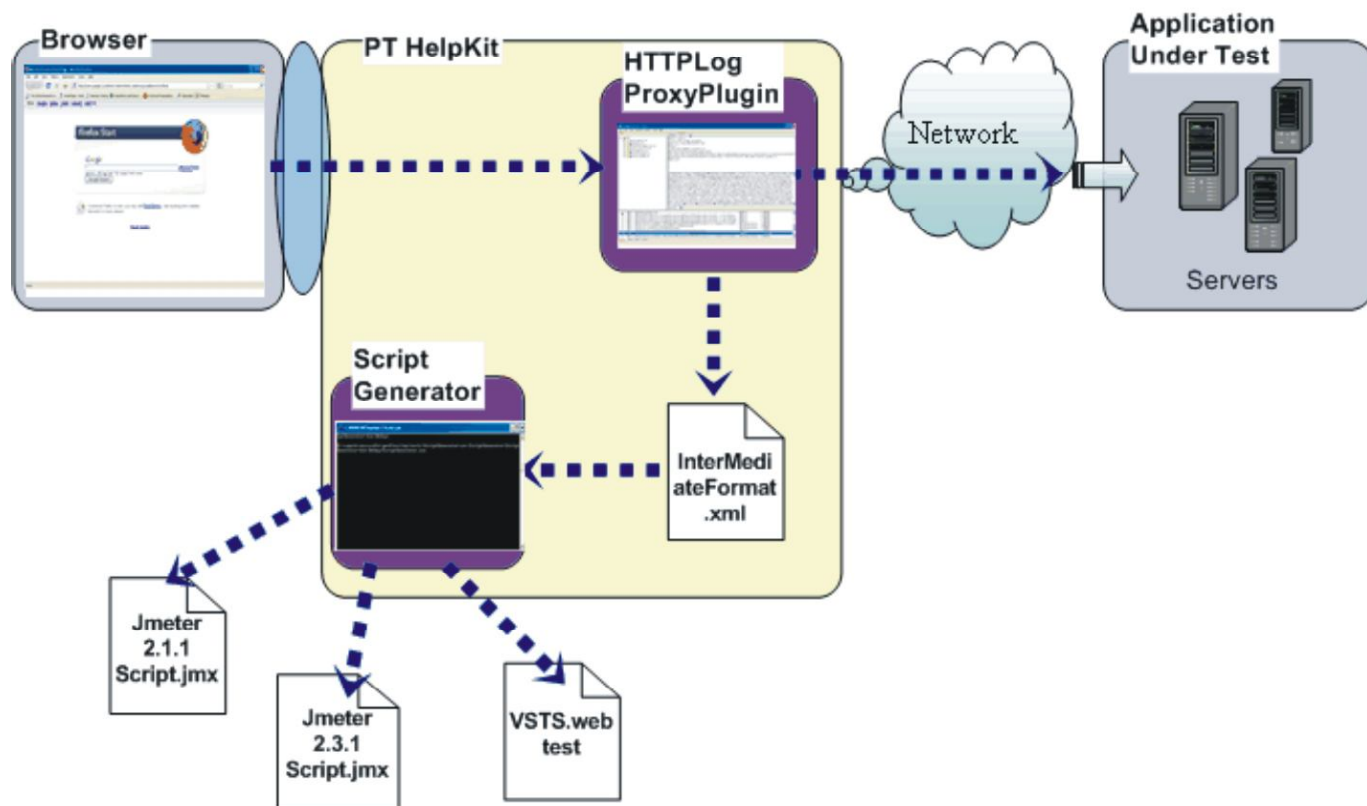
While working on web application security testing projects we had extensively used some intercepting proxies like Paros, Burp etc. Intercepting proxies can capture detailed web requests. This made us

think that if we configure each performance business scenario related web requests to pass through a proxy we could capture the required information in the raw form. So, we decided to extend one such open-source proxy tool Paros and christened the Paros extension as HTTPLog. The idea was that, if we browse the business scenario for which we want to create the script through HTTPLog (Paros proxy plug-in) we must get a processed version (IntermediateFormat) of the raw data.

Can I automatically generate a script which is specific to each PT tool?

Looking at the raw scripts of JMeter 2.1.1, JMeter 2.3.1 and VSTS 2005 it was apparent that we needed to have a generic transformation entity to convert the IntermediateFormat to a tool specific script. Thus, Script Generator was born, to play the role of the generic transformation entity which will create scripts specific to PT scripting tool.

Gradually, the PT Scripting help-kit architecture emerged, which is depicted in the diagram below:



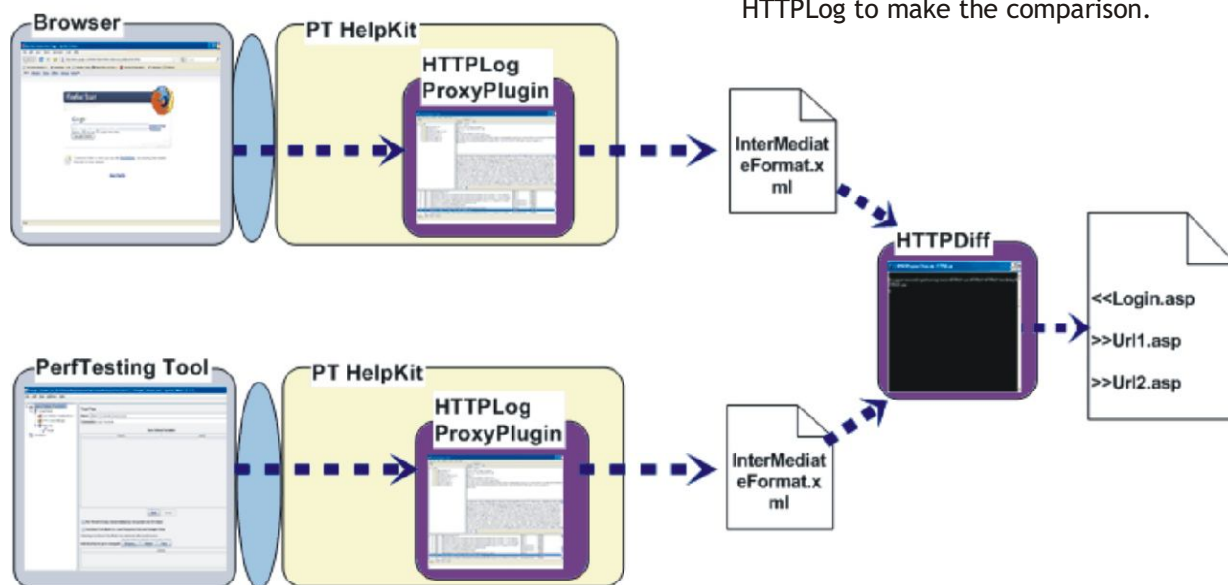
As shown in the diagram above, PT Scripting Help-kit is a collection of tools specifically designed to overcome scripting related problems faced while using open-source load generation tools.

Let's see how the PT Scripting Help-kit would fit in the scripting phase. Consider a scenario for which the user wants to create a JMeter 2.3.1 script. Before browsing the business scenario through a web browser the user needs to configure the HTTPLog (Paros proxy extension). This would cause all the information related to the current business scenario to be stored in the IntermediateFormat. Now, the Script generator would come into the picture to process the data provided in intermediate format and to create the relevant script.

The current architecture could be used to generate the scripts for commercial tools as well. Substantiating our theory and renders the scripting process completely independent of scripting tools.

How do I debug and resolve the problems in PT scripts?

The HTTPLog, IntermediateFormat and Script Generator i.e. PT Scripting Help-kit solved just the problem of script recording to an extent but debugging the scripts still remained a major hurdle. As mentioned before debugging is a complex and a very time consuming activity. Many a time, a script fails because of trivial reasons such as extra or missing parameters, improperly set parameter etc. This problem can be easily resolved if one can compare the web request fired by the load generation tool with the original web request. It is a tedious task to be done manually especially when the web requests have large set of parameters (as is the case in real life). Thus, emerged the need of a new tool which could be used to compare any two scenarios i.e. HTTPDiff. HTTPDiff feeds on the Intermediate format i.e. xml files provide by HTTPLog to make the comparison.



The diagram above shows how the PT Scripting Help-kit can also be used to debug the PT scripts. HTTPLog, IntermediateFormat and HTTPDiff can be used to find the differences in the requests fired by the load generation tool and the original requests.

Firstly, the performance engineer must browse the scenario using a web browser (configured to use HTTPLog as proxy) in order to collect the HTTP session information in Intermediate format i.e. xml file. Secondly, the performance engineer must create a script using the Script generator and run the script using a load generating tool. For a failing script the performance tester must execute the PT script (through HTTPLog) and capture the HTTP session information in another file. These two different xml files are given as input to the HTTPDiff for comparison. HTTPDiff would find the differences in terms of missing requests, extra requests etc. In other words, the output of HTTPDiff would provide a detailed list of differences in both the HTTP sessions which would greatly help in debugging the scripts.

Advantages

It's always been a challenge to implement innovative ideas and to exploit them to their potential. PT Scripting Help-kit had its fair share of nerve-wrecking challenges when it was under development. But its advantages have made the effort expended on it worthwhile. Let's take a look at the advantages of PT Scripting Help-kit:

- In the traditional approach to a performance testing engagement, scripting phase came after the planning phase as the tools to be used for recording and scripting was decided in the planning phase.

But now, because of our tool independent scripting approach, we can begin scripting phase in parallel with the planning phase. This greatly reduces the time required for a performance test cycle.

- As initial draft of the script generation process is automated, the time required to create a script is constant irrespective of the complexity.
- PT Scripting Help-kit greatly helps in improvising the estimation process as the scripting time is no longer proportional to the complexity of the application.

- The Help-kit has been built using open source tools, thus its pocket friendly.
- It has successfully overcome the problem of recording and debugging shortcomings found in open source load generation tools.

Conclusion

We have attempted to demonstrate how we can overcome challenges posed by open source tools in performance test scripting phase with the right blend of home grown tools and efficient processes. We are aware that all the challenges introduced by open source tools are not addressed by these tools. But we are confident that by shedding negativity about open source tools and with a little bit of innovative thinking performance test engineers can come up with solutions to use open source tools more effectively and efficiently.

Copyright Information

This document is the exclusive property of MindTree Limited (MindTree); the recipient agrees that they may not copy, transmit, use or disclose the confidential and proprietary information in this document by any means without the expressed and written consent of MindTree. By accepting a copy, the recipient agrees to adhere to these conditions to the confidentiality of MindTree's practices and procedures.

About MindTree

MindTree Ltd. is a global IT Solutions Company specializing in IT Services, Independent Testing, Infrastructure Management and Technical Support (IMTS), knowledge Services and Product Engineering, which comprises of R&D Services and Outsourced Product Development. MindTree partners with its clients to create a transparent, value-based relationship. Our people build innovative solutions in a wide range of technology domains that enable our customers to succeed in their business goals.



Info@mindtree.com

USA:
Tel: +1 908-604-8080
Extn. No. 2506

Asia Pacific:
Australia: +612-9025 3538
India: +91 80-26711777
Japan: +81-3-3378-6081

Singapore: +65-6323 8135
UAE: +971-50-7395894

Europe:
Germany: +49-69-3085 5092
Sweden: +46-8-501 64 044
UK: +44 20-8832 1160