Volume 6, Issue 6, June 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# An Approach To Design Network Traffic Generation Framework

**Naveen Kumar Jindal*** ION, NOKIA, Chennai, Tamilnadu, India
**Praburam P I** ION, NOKIA, Chennai, Tamilnadu, India
**Damini Ojha** FNBU, NOKIA, Chennai, Tamilnadu, India

***Abstract— Need for a traffic generation tool arises to meet the demands in load testing websites, web servers, proxy servers, etc and record various parameters to analyze their performance before they are deployed in the live network. This traffic generation tool is based on open-source platform centos[1] and uses python scripts and modules to generate traffic. This generator allows the user to generate traffic ranging from simple web browsing to downloading large files. It also makes use of the virtualization concepts which allows us to generate multiple IP address each acting as a client (or user) on a single NIC. This traffic generator not only generates traffic but also records various parameters like network speed , packets sent and received, number of active users, etc used to analyze the performance of DUT[2] at various stages of testing. It is a complete package ranging from traffic generation to monitoring the results obtained during the test. This paper also compares three methods or modules "requests[3], urllib2[4] and curl[5]" with respect to time taken in generating the same amount of traffic under the same network conditions***

***Keywords— Device under test, requests, virtual machines, urllib2, curl***

## I. INTRODUCTION

The requirements of customers are changing rapidly with the evolution of technology. Hence, it is important to load test websites and web-servers for their stability and reliability before they are deployed on the live network. It is possible that some servers are not loaded with heavy traffic during the testing phase. This means that they are not tested in an optimum manner before getting deployed which might cause the servers to fail when subjected to heavy load on the live network. This might lead to a heavy outage which can't be accepted as it results in huge economical loss. Keeping in mind all these scenarios, it becomes all the more important for traffic generation tools to generate traffic in every possible form to load test the servers. A traffic generation tool must be able to generate ample number of HTTP requests per second. This paper focuses on a tool which can be used to achieve this mark by increasing the number of IP address (clients). In other words, the number of requests per second can be scaled up to a desired level by increasing the number of IP address. A traffic generator should also generate the same number of requests in every run. There are scenarios where it has been reported that a traffic generator does not generate the same number of HTTP requests when run at different times. This results in inaccurate data capture which causes the benchmarking results to be unreliable. What the tester wants is to test the web-servers with respect to number of requests rather than time. This paper focuses on a tool which can generate traffic with respect to number of requests as well as on the basis of time. A tester can't be sure when he tests the servers with respect to time because a tool might generate different number of HTTP requests in every run. For this purpose, our tool allows the testers to test the servers or websites on the basis of number of requests rather than time. By doing so, the testers can validate their servers on request basis rather than time basis.

To make sure that a testing tool is not expensive, it becomes all the more important that it doesn't consume a lot of resources. By resources, we mean hardware. A tool must be capable of being installed on a cloud [6], so that the users do not dedicate a complete server for a testing tool alone. By doing so, they not only save money but also make sure that they use their existing hardware in an optimum manner. This paper uses the concept of *Virtual Machines[7]*. Thus this tool can also be used as a software package which can directly be installed on a cloud to load test the applications, websites, web-servers, etc.

A testing tool must emulate a lot of users, which are called virtual users. These virtual users are used to generate load. With the virtualization concept we can save a lot of computer resources. That is, we can save a lot of memory, CPU and storage space. In the absence of virtualization, one would need a dedicated computer for each user. Imagine a scenario where a web-server is to be tested with hundreds of clients generating traffic on to it. In such scenarios, creating a pool of computers becomes impossible as it is very expensive as well as the testing environment[8] doesn't have room to occupy these pool of computers. Virtual Machines come in handy in such scenarios. A virtual Machine installed with centos enables us to create multiple users on a single machine each having a different IP address. Thus, it emulates multiple users on a single computer and not only saves resources but also saves money. The ability to support creation of multiple IP address on a single NIC helps testers to generate enough load.

 

## II. RELATED WORK

In this section, we first introduce research about Software Performance Engineering (SPE) to understand how they address load generating tool's problem and show what weakpoint is. SPE represents the entire collection of activities and related analysis to meet software's performance requirements [8]. Any SPE process includes some or all of the following activities. First one is to identify and define requirement. We identify qualitative performance factors of target DUT[2] affecting performance goal [9]. And then we define detail performance factors like workload intensities, delay and throughput requirement, and scenario describing DUT behavior [10]. Second one is to predict performance. We estimate measured value of performance factors with three kinds of method like simulation-based prediction, profile-based prediction, and modeling-based prediction [11][12]. Third one is performance testing. We make load on part or all of system and measure resources of target system and performance factors. There are two categories in performance testing like load generating, for supplying the workload to a system under test, and monitor, for gathering data as the system executes. Last one is total system analysis. After making report about DUT[2] performance, we analyze the report and compare predicted performance to actual performance. Among these SPE process, we focus on load generating of performance testing.

There are many load generating tools like Load Runner [13] and Silk Performer [14]. However, tools cannot cover every testing scenario. In a server/client environment, client sends information to server generally. However, server may send information to client. Existing tools cannot support that scenario. If a tool does not understand communication protocol, it cannot generate load and if DUT[2] uses new technique that tool does not support, it cannot generate load. In those situations, we cannot use existing load generating tool and we may have to use many real computers and lot of people are required to operate software on those computers which is not economically feasible. Also, there are chances of lack of compatibility between tools and platforms on which they are being used. So, there arises a need to reduce computer resource using virtualization technology.

There are many traffic generators available. Most of the existing traffic generating tools are one-dimensional. Some of them are focused only on traffic generation. They are not focused on utilizing the hardware resources efficiently. For example, a tool like Jmeter has a lot of dependencies that should be present on the server where we are trying to install a traffic generating tool. Thus, we need to have enough storage space available on the hardware on which we are trying to install such tools. There are certain tools which are not compatible with certain operating systems. An ideal tool must be compatible with all the operating system be it Windows, Linux or any other open source platform. A tool must not only be compatible with operating system but also with all the web browsers so that they can be installed as a plug-in which would save a lot of hardware resources. A tool must also be available in the form of a software package. A summary of selected tools is provided in the following table. The basic idea of this overview is to show needed dimensions of configurability and application.

Table I: Comparison of Existing Tools

| Toolkit | Packet definition | Traffic model | Parallel operation | Packet types |
|---|---|---|---|---|
| **ttcp** | No | Through-put | No | TCP, UDP |
| **NetPerf** | No | Through-put | No | TCP, UDP |
| **NetSpec** | Limited | Variable | Yes | TCP, UDP |
| **Iperf** | No | Though-put | Yes | TCP, UDP |
| **DBS** | Limited | Variable | Yes | TCP, UDP |
| **Harpoon** | No | Variable | Yes | TCP, UDP |
| **npag** | Yes | Variable | Yes | TCP,UDP, ICMP, user-defined |

## III. PROPOSED APPROACH

Our proposed approach consists of two parts: architecture and selection of virtualization product. Architecture explains the flow of execution of the tool and selection of virtualization product deals with the software entity which makes use of the available hardware resources effectively. Virtualization makes our tool economically feasible. Virtualization also makes sure that this tool can be implemented on cloud.

The overall architecture of the tool is presented in Figure 1. Virtual computers are generated from real server. The number of virtual computers generated from a real server depends on the scenario under which the DUT[2] is to be analyzed.

 

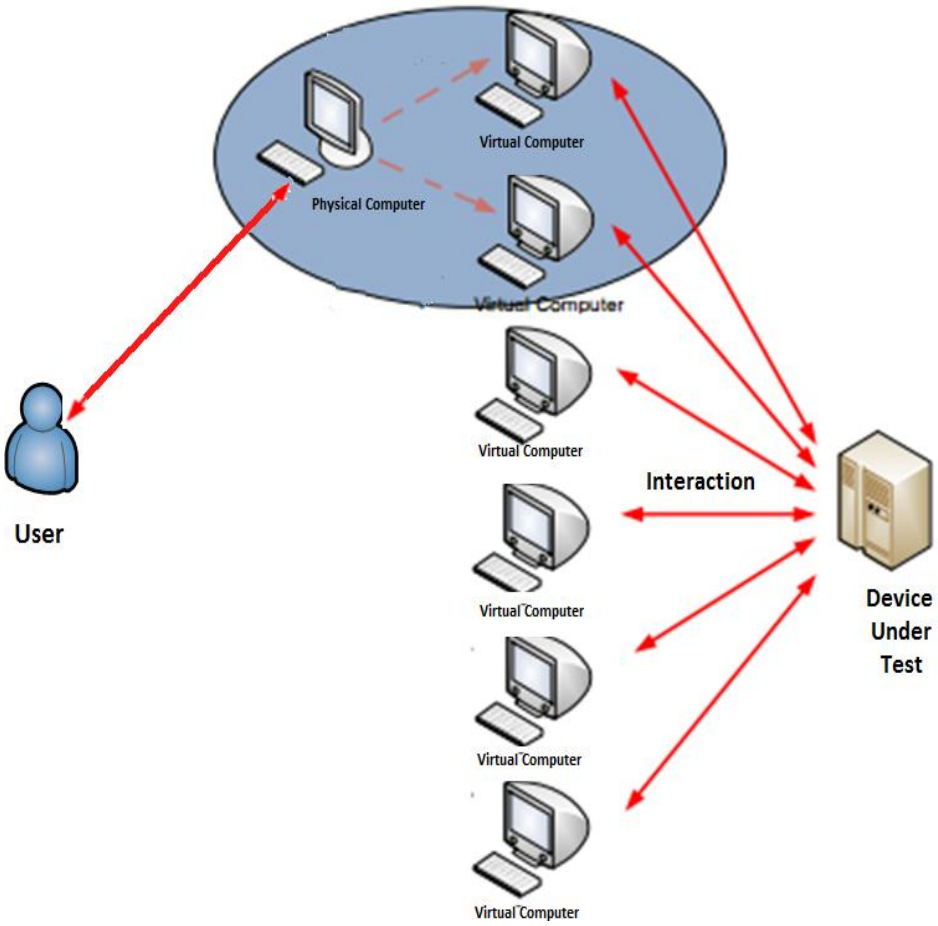


Figure 1. Architecture of the tool

Each Virtual Computer has a different IP Address. Thus, each virtual computer acts as an individual client. The user (tester) just interacts with the physical computer and executes the python scripts present. These scripts activate the virtual computers which interact with the DUT[2] and generate traffic depending on the profile which the user selects before the test begins. Virtualization is a technique that abstracts computing resource. It enables the users to create a simulated environment and use the resources effectively. Virtualization is divided into several categories as purpose such as server virtualization, network virtualization, desktop virtualization, application virtualization and so on [15]. In this paper, we use desktop virtualization technique to perform software performance test. There are a lot of desktop virtualization products such as VMWARE [16], Virtual PC [17], Parallels [18], Virtual Box [19] and so on. We selected VMWARE because it is the most frequently used product and it uses computing resource efficiently. It enables the users to use all the CPU cores effectively. It also has high compatibility with various computing devices and operating system.

## IV. OBJECTIVE

The primary objective of this traffic generating tool is to overcome the disadvantages or missing functionalities of other existing tools. Most of the existing tools, are built on the same idea. Thus, there is a very little difference in these tools. We analyzed the existing tools and concluded that a testing tool must have the following functionalities.

a. Variable traffic rates that can be used to simulate different test scenarios (Mobile and Fixed line Testing)
b. Long and short term measurements
c. Quality of service measurements (Packet loss, delays, etc)
d. Support for parallel traffic generation from a single as well as multiple user(s)
e. Flexible configuration scheme which can be manipulated according to the testing scenario.
f. A mechanism for monitoring the test results which helps the testers to analyze the results.
g. A request based model which helps in benchmarking DUT[2] on number of requests handled rather than time. By doing so, the DUT[2] can be benchmarked effectively as it is analyzed for the same parameters after every test.
h. An easy to access GUI which helps the testers to configure various parameters easily.

## V. DESIGN

In a distributed manner, this tool is used to generate traffic at multiple locations in the network in order to simulate observable network traffic (See Figure 2). For example, DDoS( Distributed Denial-of-Service) attacks can be formulated and executed.

 

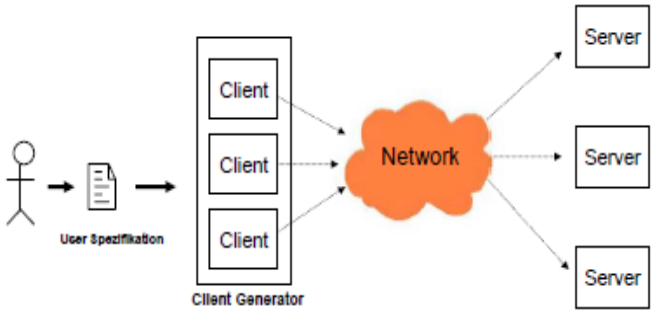


Figure 2. Design of Working Model

This tool can test a number of servers at the same time as shown in Figure 2. The design of this tool is based on abstractions and paradigms. Basically, a client-server model is established which allows formulation of TCP connections. Before the traffic is generated, a TCP session is established between each client and the server. After the TCP sessions are established , network packets are exchanged between the server and the clients. This exchanging of network packets is nothing but traffic generation. Traffic generation can be sequential as well as parallel. Most of the DUT[2] must be tested under parallel traffic because only then their resources are tested effectively. This tool supports parallel traffic generation by using *threading* concepts. A simple HTTP request is created as a small thread. After the creation of threads, they are called concurrently one after the other thus generating parallel traffic. The main objective of this tool is to support parallel traffic streams from multiple IP address at the same time. This makes sure that the DUT[2] is load tested in an optimum manner before it is deployed on the live network. This tool also presents a GUI[20] for the end-user which has a variety of options ranging from number of clients, number of servers, type of traffic and the file size to be requested for. It also supports quality of service (QOS) by providing a method to ramp up the traffic up to its maximum limit after a certain period of time. It also has a support to capture the *packet trace* which can be used in analyzing the packet flow which helps in troubleshooting in the case of network failure. . Most of the existing tools do not support monitoring and rely on other methods like *nmon, graphite, nload, etc.* This tool also has a mechanism to record various parameters like CPU-usage, Speed at which packets are sent and received, total number of packets sent and received, response-time of the DUT, etc.These parameters are represented in a graphical manner which enables the testers to analyze the DUT performance under different kinds of loads.

## VI. OBSERVATION

We use three modules “requests, urllib2 along with httplib and linux curl” to generate traffic. All these modules are open source and can be downloaded from python and linux repositories. We found that all these modules support parallel traffic generation. This section covers the functionality of each module and compares them on trivial features like the time taken to generate the same amount of traffic under the same network conditions. It was observed that urllib2 module and linux-curl support multiple IP addresses whereas requests module does not. It was also observed that urllib2 module is the fastest in terms of generating 10000 HTTP requests from a single IP as well as multiple IP address or clients. The following table summarizes the three modules used in this tool.

| Module | Number Of HTTP Requests | Time Taken by Single Client | Time Taken by Multiple Clients (10) |
|---|---|---|---|
| Requests | 10000 | 32 Seconds | Doesn’t support Multiple IP Feature |
| Urllib2 | 10000 | 25 Seconds | 20 Seconds |
| Linux-Curl | 10000 | 48 Seconds | 40 Seconds |

## VII. CONCLUSION AND FUTURE WORKS

It is important to save human resources and computer resources to make a tool economically feasible. In this paper we presented a tool which is based on python and makes use of the virtualization technology. Thus, we focused on generating traffic in an economical manner and suggested ways of saving hardware resources by making use of the virtualization concepts. We made use of three python modules to generate HTTP traffic and compared them on the basis of time taken to generate the same number of HTTP requests. In the future, we plan to include few more performance monitoring parameters and use the multi-processing feature which provides testers with a way to control the number of TCP sessions and HTTP Connections which are established.

 

**REFERENCES**

[1] Compuware, "Applied Performance Management Survey", Oct, 2006

[2] M. Woodside, G. Franks, D. Petriu, "The Future of Software Performance Engineering", SOSE'07, pp. 171-187.

[3] L. Chung, B. A. Nixon, E. Yu, J. Mylopoulos, "NonFunctional Requirements in Software Engineering", Kluwer, 2000.

[4] S. Barber, "User Community Modeling Language for performance test workloads"

[5] G. Zheng, T. Wilmarth, P. Jagadishprasad, and L. V. Kalé, "Simulation-Based Performance Prediction for Large Parallel Machines", International Journal of Parallel Programming, Vol. 33, Num. 2-3, 2005.

[6] S.Balsamo, A. DiMarco, P. Inverardi, and M. Simeoni, "Model-based Performance Predition in Software Deveolopment", IEEE Trans. On Software Eng, vol 30, 2004, pp. 295-310.

[7] T. Kanade, P. Rander, P.J. Narayanan , "Virtualized reality: constructing virtual worlds from real scenes", IEEE Multimedia, Immersive Telepresence, Vol. 4, No. 1, January, 1997, pp. 34-47

 