x

Evaluation of Open Source Tools for Application Software Testing over Commercial Tools

Submitted By

Akbari Hardi M.

13mcei01

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2015

Evaluation of Open Source Tools for Application Software Testing over Commercial Tools

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Akbari Hardi M.

(13mcei01)

Guided By

Prof. Zunnun Narmawala (Internal) Mr. Deepak Khatri (External) Ms. Anna Maria Rosario (External)

NIRMA
UNIVERSITY
INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2015

Certificate

This is to certify that the major project entitled "Evaluation of Open Source Tools for Application Software Testing over Commercial Tools" submitted by Akbari Hardi M. (Roll No: 13cei01), towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Zunnun Narmawala
Guide & Assistant Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Prof. Sharada Valiveti
Associate Professor,
Coordinator M.Tech - INS
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr K Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

|||||||||||||||||||||||||||||||||||||||

I, Akbari Hardi M., Roll. No. 13mcei01, give undertaking that the Major Project entitled "Evaluation of Open Source Tools for Application Software Testing over Commercial Tools" submitted by me, towards the partial ful llment of the requirements for the degree of Master of Technology in Computer Science & Engineering of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

|||||||{

Signature of Student

Date:

Place:

Endorsed by Prof.
Zunnun Narmawala
(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to Prof. Zunnun Narmawala, Assistant Professor, Computer Science Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank Dr. Sanjay Garg, Hon'ble Head of Com-puter Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to Dr K Kotecha, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank Mr. Deepak Khatri and Ms. Anna Maria Rosario for their special attention and suggestions towards the project work.

Nevertheless, I express my gratitude toward my family, friends and colleagues for their kind co-operation and encouragement which help me in completion of this project.

- Akbari Hardi M.
13mcei01

Abstract

A software developing organization undergoes various phases of development. Our company, ST Microelectronics plays a vital role in software development and the same falls true for it too. Testing is one of the most important phases and our dissertation has been pursued in correlation to that. We have mainly paid attention on Security testing and Performance testing.

We have surveyed various static source code analyzers. We have investigated issues associated with them. Our dissertation concerns in development of open source software. The existing licensed ones are found to be expensive and unable to be used in many circumstances. Our proposed work has replaced them with feasible and cost effective open source software which have met all the requirements of the existing ones. AppScan, YASCA, FindBugs and RATS are tools which have been researched thoroughly and a feasible solution in accordance to expense and security has been proposed and received.

The achieved results have been used further for perfromance testing. We have also proposed a solution which can achieve TMMI level 3. LoadRunner, Jmeter, locust, LoadTester and BlazeMeter tools have been studied too. The whole analysis has given the most optimal solution, useful for replacing expensive licensed versions of tools with a cost effective alternative approach which fulfils both, security and performance testing.

Abbreviations

TMMI	Test Maturity Model Integration
PT	Performance testing

||||||||||||||||||||||||||||||||||||||

{

Contents

List of Figures

List of Tables

Chapter 1

Introduction

STMicroelectronics is the world's 5th largest semiconductor company with net revenues of US $11.51 billion in 2013. ST serves customers across the spectrum of electronics applications with innovative semiconductor solutions by leveraging its vast array of tech-nologies, design expertise and combination of intellectual property port- folio, strategic partnerships and manufacturing strength. STMicroelectronics was created in 1987 by the merger of SGS Microelectronic of Italy and Thomson Semi- conductors of France with the aim of becoming a world leader in the sub-micron area.[4]

1.1 Knowledge Discovery Process

Performance testing is by and large testing performed to decide how a application per-forms as far as responsiveness and strength under a specific workload. It can likewise serve to explore, measure, accept or check other quality traits of the system, for example, adaptability, dependability and asset utilization.

Performance testing is a subset of Performance engineering and is a software engineering practice which endeavors to incorporate execution with the usage, outline and building design of a application or system to be tested. [5], [6] , [7]

1.2 Problem Statement

Testing is one of the important phases of SDLC. Performance testing is genuinely viewed as a standout amongst the most in fact complex sorts of programming testing, on the grounds that it obliges testers on broad to have specialized information and involvement in programming. Having such a test, you will find out about the bottlenecks in your application and in system, focus for yourself the at scaling it with the quantity of clients and get elaborated proposals to enhance execution.

An effective performance testing will extend the greater part of the execution issues, which could be identified with database, system, programming, equipment and so on. The essential objective incorporates making the benchmark conduct of the application tested. There are various industry-characterized benchmarks, which ought to be met. Performance testing does not plan to discover abandons in the application accordingly. It really addresses some more basic errand of testing the benchmark and the norms set for the application. Exactness and close observing of the execution and consequences of the test is the essential normal for execution testing.[8] [9]

At ST Micro security testing of all the applications is done using HP Load runner which is licensed software. It costs much and company wants to cut cost so we try to analyze other options available which could provide all the required functionality. We tried to compare the feasible options and then demonstrated its pros and cons. We proposed feasible solution and also added the missing features.

1.3 Objective of Study

The main objective of this study is to suggest feasible solution to licensed software used for performance testing which could be cost effective as well as functionally proper. After researching on all the options available for the same, a detailed study on 3 tools is done in order to compare with presently used tools.

1.4 Scope of Work

In this study we focus on performance testing only, other types of testing is not focused. Program analysis is the process of automatically analyzing the behavior of computer programs. Two main approaches in program analysis are static program analysis and dynamic program analysis. This work focuses on analysis of tools that could help the organization.

Chapter 2

Literature Survey

Literature Survey is an important aspect in the development of any project. While working on testing, we have come across new terms and various concepts. These have played a vital role in the growing phase of our studies and while working on it. Below is the brief description of our literature survey that has built a strong base:

2.1 Literature Review

Performance testing, inside organization, includes the roles, activities, tools, practices and deliverable applied at each period of the SDLC and guarantees that an answer will be composed, actualized, and operationally upheld to meet the non-utilitarian necessities for execution, (for example, throughput or memory use). It might be then again alluded to as operational efficiency or application efficiency tuning inside programming of the application. As the association between application achievement and business achievement keeps on picking up acknowledgment, especially in the portable space, application efficiency tuning has tackled a safeguard and perfectible part inside the product advancement life cycle. Thus, the term is commonly used to portray the courses of action, individuals and team needed to adequately test non-useful necessities, guarantee adherence to administration levels and upgrade application execution before deploying the application. The term efficiency envelops more than simply the product and supporting foundation, and in that capacity the term operational efficiency is ideal from a large scale view. Adherence to the non-utilitarian necessities is additionally approved post-organization by checking the creation of applications. Operational efficiency, using Performance testing, has turned into a different control at various organizations, with tasking separate yet parallel to Systems Engineering. It is pervasive, including individu-als from various hierarchical units; yet overwhelmingly inside the ICT department. [4] , [?] , [9] , [?]

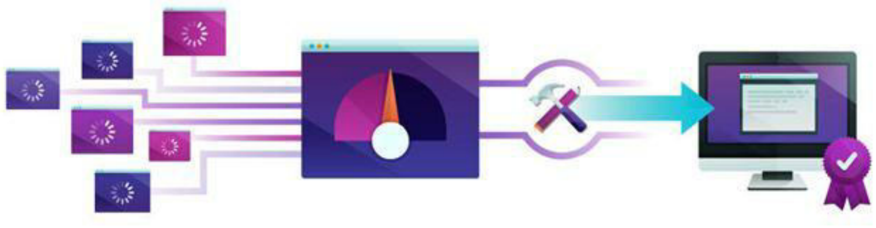


Figure 2.1: Performance testing

2.2 Performance Testing Types [1] , [2], [3]

Load testing: A load test is normally led to comprehend the conduct of the application under a particular expected burden. This can be the normal simultaneous number of clients on the application performing a particular number of exchanges inside the set span. This test will give out the reaction times of all the critical busi-ness discriminating exchanges. In the event that the database, application server, and so forth are likewise checked, then this basic test can itself point towards bot-tlenecks in the application programming.

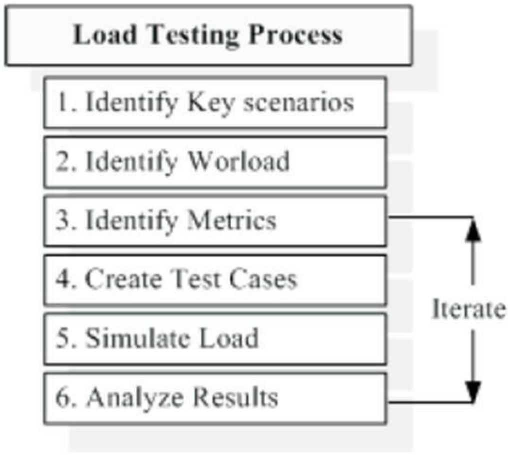


Figure 2.2: Load Testing Process

Stress testing: It is typically used to comprehend the maximum furthest reaches of limit inside the application. This sort of test is done to focus the application framework's power regarding great load and helps application executives to figure out whether it will perform adequately if the present burden goes well over the normal tested.[10]

Soak testing It is also known as endurance testing. It is generally done to g-ure out whether the framework can maintain the consistent expected burden. Amid tests, memory use is checked to distinguish potential holes. Likewise critical, yet fre-quently disregarded is execution debasement, i.e. to guarantee that the throughput and/or reaction times after some long stretch of supported movement are tanta-mount to or better than toward the start of the test. It basically includes applying a huge burden to a framework for an amplified, critical time of time. The objective is to find how the framework carries on under supported utilization.

Spike testing It is finished by abruptly expanding the heap created by an extensive number of users, and watching the conduct of the application. The objective is to figure out if execution will endure, the application will fall at, or it will have the capacity to handle emotional changes in burden.

Configuration testing As opposed to testing for execution from a heap viewpoint, tests are made to focus the impacts of design changes to the application framework's parts on the framework's execution and conduct. A typical illustration would be trying different things with distinctive routines for burden adjusting.

Isolation testing Separation testing is not unique to testing but rather includes rehashing a test execution that brought about a framework issue. Such testing can frequently detach and affirm the deficiency space.[11]

2.3 Comparison of Tools

After a detailed discussion with team members it was decided to compare following tools: After final discussion with expert, it was deduced to do detailed study of 4 tools viz Jmeter, locust, LoadTester and BlazeMeter.

Table 2.1: Comparison of Tools

Name of tool	Platform	License type
Load Runner	Windows	Commercial
Jmeter	Windows, Linux	Open source
AgileLoad	Windows	Open source
locust	Linux	Open source
LoadTester	Windows	Freemium
WAPT	Linux	Open source
PyLOT	Windows	Open source
BlazeMeter	Windows	Commercial

Chapter 3

Comparative Study of Tools

3.1 Load Runner

LoadRunner is a product testing apparatus from Hewlett-Packard. It is utilized to test applications, measuring application conduct and execution under burden.HP LoadRunner can reenact a large number of clients simultaneously utilizing application programming, recording and later investigating the execution of key parts of the application. LoadRun-ner reproduces client action by producing messages between application parts instead of reenacting collaborations with the client interface, for example, keypresses or mouse developments. The messages to be created are put away in scripts. LoadRunner can create the scripts by recording them, for example, logging HTTP appeals beween a customer web program and an application's web server. [12] [13]

3.1.1 Architecture

[14] , [15] , [16] The key segments of HP LoadRunner are: Load Generator produces the load against the application by taking after scripts.

VuGen (Virtual User Generator) for creating and altering scripts

Controller controls, dispatches and groupings occasions of Load Generator - determining which script to use, for to what extent and so on. Amid runs the Controller gets constant checking information and showcases status.

Agents procedure oversees association in the middle of Controller and Load Gener-ator occasions.

Analysis collects logs from different burden generators and organizations reports for visualization of run result information and checking information.

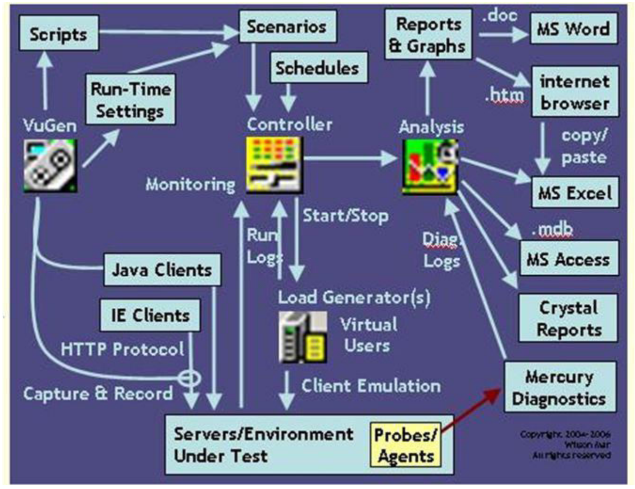


Figure 3.1: Load Runner Architecture

3.1.2 Working:

Load Runner works by making virtual clients who take the spot of genuine clients working customer programming, for example, Internet Explorer sending solicitations utilizing the HTTP convention to IIS or Apache web servers. Demands from numerous virtual client customers are created by "Load Generators" with a specific end goal to make a heap on different servers under test.

These load generator specialists are began and halted by the "Controller" project. The Controller controls burden test runs in light of "Scenarios" conjuring arranged "Scripts" and related "Run-time Settings". Scripts are created utilizing the "Virtual user script Generator" (named "V U Gen"), It creates C language script code to be executed by virtual clients by catching system movement between Internet application customers and servers. With Java customers, VuGen catches calls by snaring inside the customer JVM.

Amid runs, the status of every machine is checked by the Controller. Toward the end of every run, the Controller consolidates its checking logs with logs acquired from burden generators, and makes them accessible to the "Analysis" program, which can then make run result reports and diagrams for Microsoft Word, Crystal Reports, or a HTML page program.

Every HTML report page created by Analysis incorporates a connection to results in a content record which Microsoft Excel can open to perform extra investigation. Error amid every run is put away in a database. [17] [13]

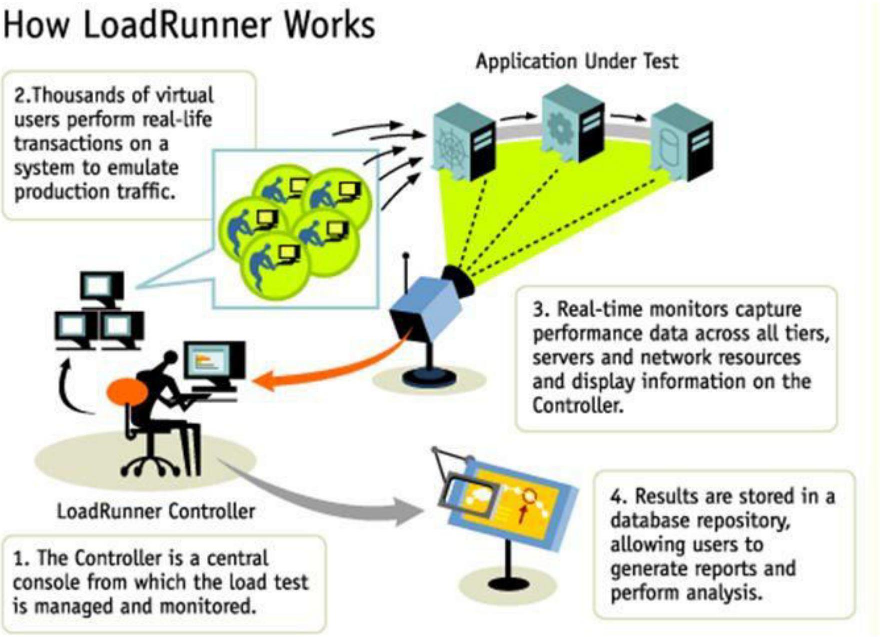
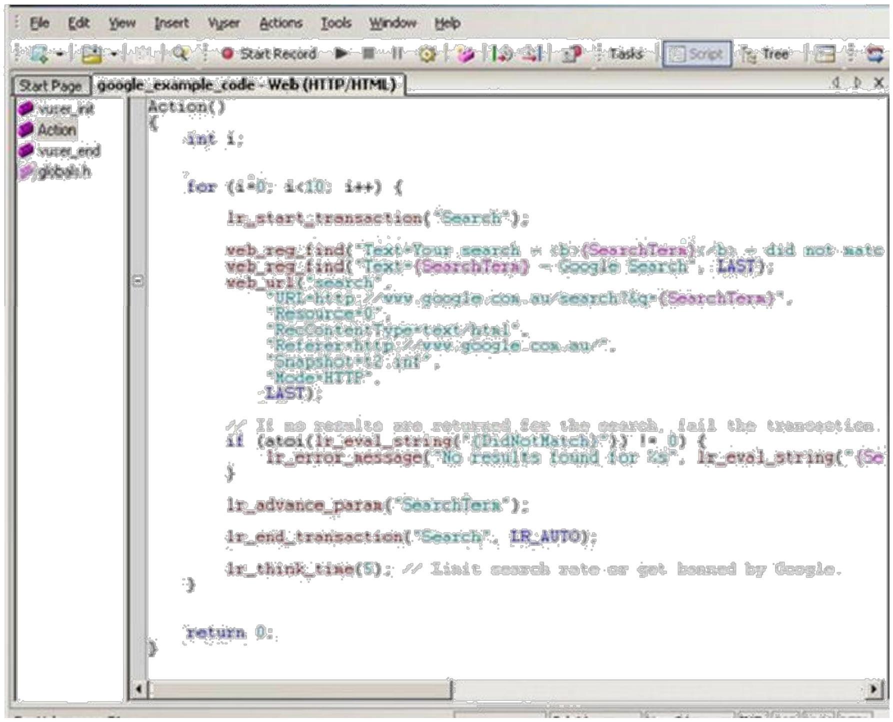

How LoadRunner Works
Application Under Test
2.Thousands of virtual users perform real-life transactions on a system to emulate production traffic.
3. Real-time monitors capture performance data across all tiers, servers and network resources and display information on the Controller.
LoadRunner Controller
1. The Controller is a central console from which the load test is managed and monitored.
4. Results are stored in a database repository, allowing users to generate reports and perform analysis.

Figure 3.2: Load Runner Working

Following screen shows working:


File Edit View Insert Vuser Actions Tools Window Help
Start Record
Tasks
Script
Tree
Start Page
google_example_code - Web (HTTP/HTML)
vuser_init
Action
vuser_end
globals.h

```
Action()
{
    int i;

    for (i=0; i<10; i++) {

        lr_start_transaction("Search");

        web_reg_find("Text=Your search - <b>{SearchTerm}</b> - did not matc
        web_reg_find("Text={SearchTerm} - Google Search", LAST);
        web_url("search",
            "URL=http://www.google.com.au/search?&q={SearchTerm}",
            "Resource=0",
            "RecContentType=text/html",
            "Referer=http://www.google.com.au/",
            "Snapshot=t2.inf",
            "Mode=HTTP",
           LAST);

        // If no results are returned for the search, fail the transaction.
        if (atoi(lr_eval_string("{DidNotMatch}")) != 0) {
            lr_error_message("No results found for %s", lr_eval_string("{Se
        }

        lr_advance_param("SearchTerm");

        lr_end_transaction("Search", LR_AUTO);

        lr_think_time(5); // Limit search rate or get banned by Google.
    }

    return 0;
}
```

Figure 3.3: Load Runner VUGen

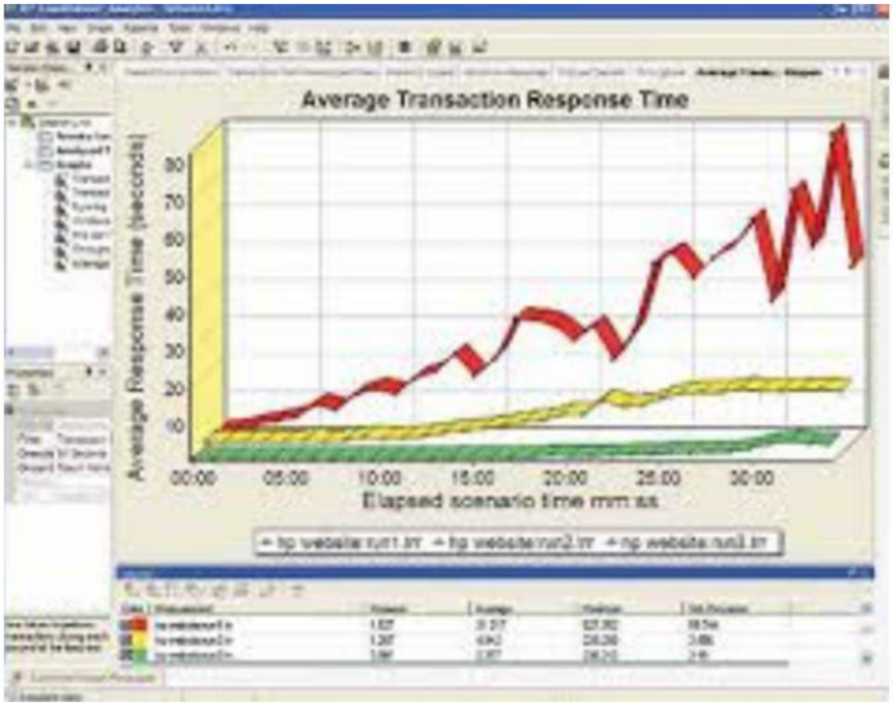


Figure 3.4: Load Runner Analysis

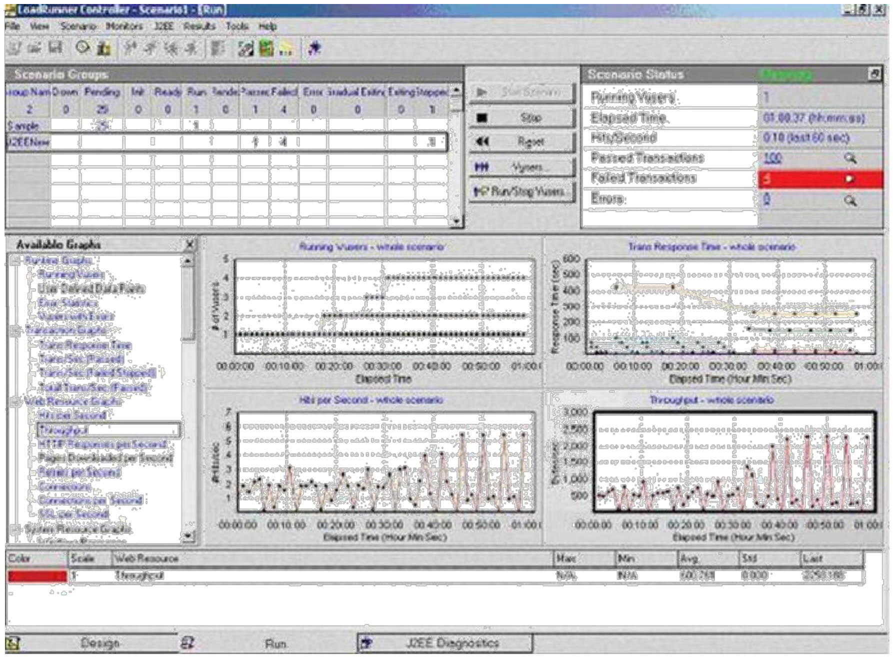

LoadRunner Controller - Scenario1 - [Run]
File View Scenario Monitors J2EE Results Tools Help
Scenario Groups
Stop
Reset
Vusers...
Run/Stop Vusers...
Scenario Status
Running Vusers
Elapsed Time
Hits/Second
Passed Transactions
Failed Transactions
Errors
Available Graphs
Running Vusers - whole scenario
Trans Response Time - whole scenario
Hits per Second - whole scenario
Throughput - whole scenario
Elapsed Time
Design
Run
J2EE Diagnostics

Figure 3.5: Load Runner Controller

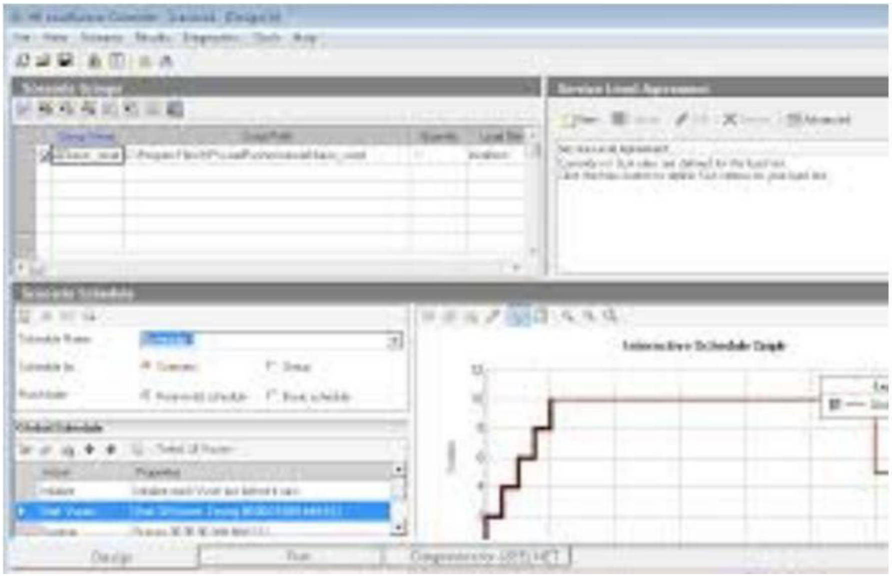


Figure 3.6: Load Runner Controller Design

3.1.3 Test results

Following table shows results evaluated after execution of 43 applications (20 java, 16 .net and 7 php based) for this tool: NOTE:values are considered for 2 decimal points only

Virtual User	Technology of Application	Response Time (sec)	Memory Utilization (MB)	Success Rate (%)	Error Rate
1	Java	0.001	2	100	0
10		0.01	18.8		
100		0.1	179.0.		
1	.Net	0.002	5	100	0
10		0.2	48.7		
100		2.034	583.76		
1	php	0.016	3.45	100	0
10		0.189	33.23		
100		2.002	289.69		

Table 3.1: Table

3.2 Jmeter

Apache JMeter is open source programming, a 100% unadulterated Java desktop appli-cation intended to load test utilitarian conduct and measure execution. It was initially intended for testing Web Applications however has subsequent to extended to other test capacities. Apache JMeter is an Apache extend that can be utilized as a load testing ap-paratus for dissecting and measuring the execution of an assortment of administrations, with an attention on web applications. JMeter can be utilized as a unit test instrument for JDBC database connections: FTP, LDAP, Webservices, JMS, HTTP. JMeter can likewise be arranged as a monitor, despite the fact that this is regularly viewed as a specially appointed arrangement in lieu of cutting edge observing arrangements. JMeter offers variable parametrization, every string treats, arrangement variables and a mixed bag of reports. [18] [19]

3.2.1 Archietecture and working

JMeter is written in the Java, with a produced Javadoc. Jmeter comprised of a Master system (the Jmeter GUI) which controls remote slave frameworks running jmeter-server examples which simultaneously forces stack on a target server, system or protest under test by copying movement to and from customer programming.

The JMeter GUI (ApacheJmeter.jar) is a multi-strung Java class running Java Swing interfaces. It is conjured utilizing jmeter.bat. jmeter-server speaks with different remote injector Java RMIRegistry administrations. Remote servers as a matter of course listens to port 1099. [20] [21]

A Test Plan is a compartment for components which indicates the parameters for test runs. Config. components and Listeners can be on any level. Samplers store screen server measurements into .jtl (JMeter Test Log) documents.

Every Thread Group reproduces an individual virtual client. Every string is a unit of work that can be executed all the while or consecutively. To every string gathering can be included Logic Controllers and Elements.

JMeter has a multi-threaded structural planning that empowers Java engineers to augment JMeter with custom plugins and usefulness augmentations, recorded at code.google.com/p/jmeter-plugins/, depicted here.

JMeter presents diagrams of run results (end-to-end execution under load over time). [18] [22] [21]

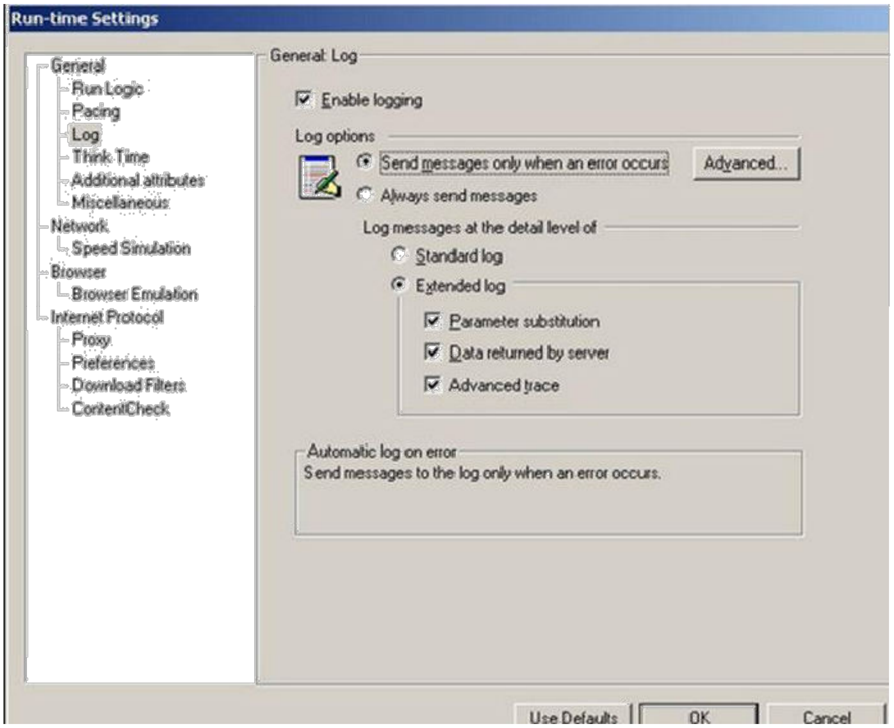
Following screen shows working:

3.2.2 Test results

Following table shows results evaluated after execution of 43 applications (20 java, 16 .net and 7 php based) for this tool: NOTE:values are considered for 2 decimal points only

Virtual User	Technology of Application	Response Time (sec)	Memory Utilization (MB)	Success Rate (%)	Error Rate
1 10 100	Java	0.001 0.01 0.1	2 21.2 199.09	98.2	1.8
1 10 100	.Net	0.002 0.2 2.89	4.34 50.3 469.09	100	0
1 10 100	php	0.02 0.19 2.09	2.42 31.08 272.69	99	1

Table 3.2: Table



Run-time Settings

- General
  - Run Logic
  - Pacing
  - Log
  - Think Time
  - Additional attributes
  - Miscellaneous
- Network
  - Speed Simulation
- Browser
  - Browser Emulation
- Internet Protocol
  - Proxy
  - Preferences
  - Download Filters
  - ContentCheck

General: Log

☑ Enable logging

Log options

● Send messages only when an error occurs [Advanced...]

○ Always send messages

Log messages at the detail level of

○ Standard log

● Extended log

☑ Parameter substitution

☑ Data returned by server

☑ Advanced trace

Automatic log on error

Send messages to the log only when an error occurs.

Figure 3.7: Load Runner Runtime Settings

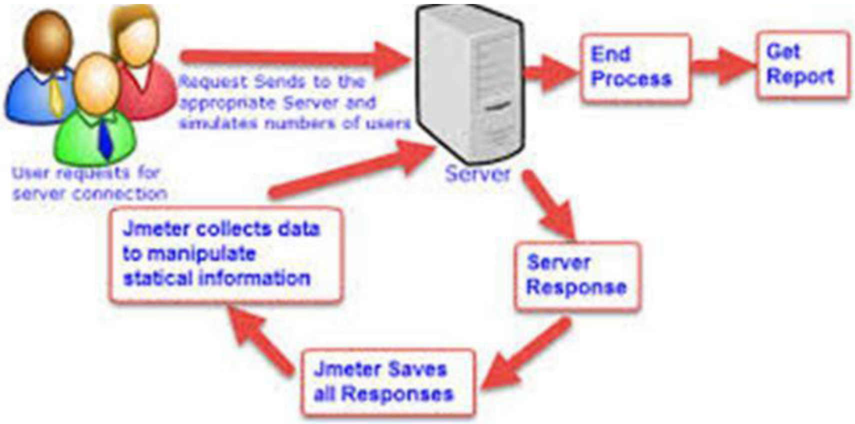

Request Sends to the appropriate Server and simulates numbers of users
Server
End Process
Get Report
User requests for server connection
Jmeter collects data to manipulate statical information
Server Response
Jmeter Saves all Responses

Figure 3.8: JMeter Archietecture

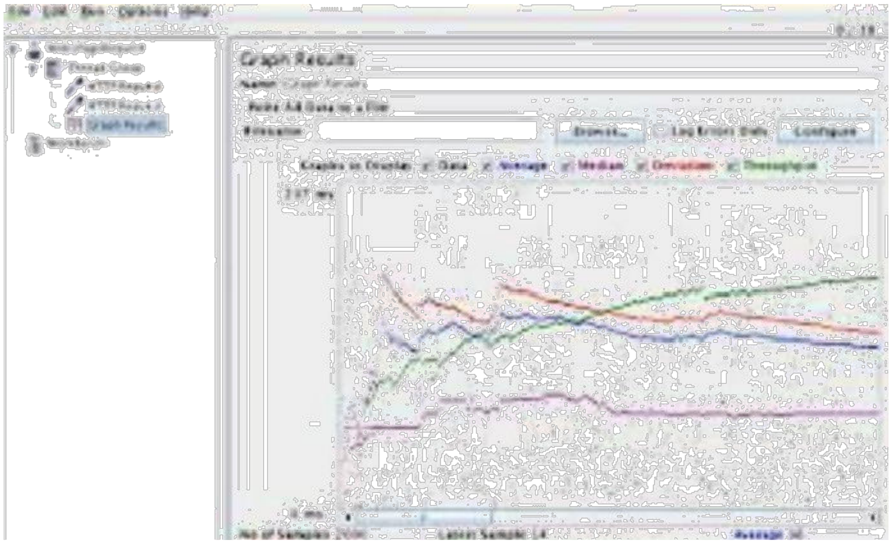


Figure 3.9: JMeter Analysis

3.3 Locust

It is an open source load testing tool written in Python. It permits you to characterize client conduct with Python code, and swarm your framework with many concurrent users. It is a simple to-utilize, appropriated, client burden testing device. Planned for burden testing sites((or different frameworks) and making sense of what number of simultaneous clients a framework can deal with. [23]

The thought is that amid a test, a swarm of users will attack application. The conduct of every test client is characterized and the swarming methodology is observed from a web UI progressively. This will help to test and recognize bottlenecks in code before letting genuine clients in ie. before deploying.[24] , [25] , [6]

3.3.1 Architecture and working

The execution stream begins in the primary capacity of main.py. Other alternatives that it was begun with are parsed. The locust file that characterizes the test is stacked and Locust classes with related "tasks" are parsed and put away. Assignments are just adorned capacities that perform HTTP asks for, for example, GET and POST.

To do the actual HTTP requests the python standard library urllib2 is used. The browser that is used by the simulated users is defined in clients.py. When a request is executed the time until a response has been received is recorded. It is implemented using events. When a request is finished, an event is fired to allow the stats module to store a response time or possibly a failure.

Every recreated clients run in Greenlet strings. A greenlet is really not a genuine string but rather sense well to consider it such. The greenlets are planned (brought forth and slaughtered) by a runner in the runner beetle module. The runner it self is a greenlet produced from the fundamental module. The runner can request its recreated clients to begin (bringing forth greenlets) or to quit (slaughtering greenlets). At the point when a mimicked client has been begun its nearby greenlet handles the real demands and holding up.

Following screens show working:

3.3.2 Test results

Following table shows results evaluated after execution of 43 applications (20 java, 16 .net and 7 php based) for this tool: NOTE:values are considered for 2 decimal points only

Virtual User	Technology of Application	Response Time (sec)	Memory Utilization (MB)	Success Rate (%)	Error Rate
1 10 100	Java	2.05 19.49 0.1	3 20.72 209.74	100	0
1 10 100	.Net	0.002 0.2 2.034	3.78 19.95 222.65	89.4	10.6
1 10 100	php	0.016 17.04 264.84	4.03 23.75 207.59	92	8

Table 3.3: Table

3.4 LoadTester

Web Performance Load Tester is Freemium for testing load on windows platform. It is only web testing tool savvy enough to let you know what number of clients your site can deal with. Use of another instrument to outline a considerable length of time attempting to make sense of it in the event that you can even tell by any means. [26] , [27]

At the push of a button you can create load from outside your system to test the whole application stack, including the firewall, or produce load from inside your test lab to focus on server execution without anyone's input. [28]

3.4.1 Architecture and working

Web Performance LoadTester for the most part perform either load testing or API testing however not both. Then again, it can be arranged to test the whether every call meets expectations (Functional & QA Testing), and the calls velocity.

The accompanying segment contains it's working four sections alongwith screens [29]:

recording an API call using

browser making a dataset

altering the datasource

producing an arrangement of calls utilizing a custom dataset

3.4.2 Test result

Following table shows results evaluated after execution of 43 applications (20 java, 16 .net and 7 php based) for this tool: NOTE:values are considered for 2 decimal points only

Virtual User	Technology of Application	Response Time (sec	Success Rate (%)	Error Rate
1 10 100	Java	0.001 0.01 0.1	100	0
1 10 100	.Net	0.002 0.2 3	100	0
1 10 100	php	0.02 1.02 3.55	100	0

Table 3.4: Table

3.5 BlazeMeter

Blazemeter is windows based commercial testing tool. BlazeMeter is a self-administration load testing platform-as-a-service (PaaS), which is used for execution testing structure. BlazeMeter gives an undertaking evaluation, 'out-of-the-case' burden testing answer for the designer community. [30] , [31]

3.5.1 Working

BlazeMeter gives testers with instruments to a basic reconciliation into their local improvement environment by giving versatile, web application, site, web-administration or database testing that can reproduce many users who are going to a site simultaneously utilizing the administration. [32] , [33]

Users can run numerous load tests keeping in mind the end goal to find and fix execution bottlenecks. BlazeMeter's load trying stage has fabricated in incorporations that can be stretched out with a progression of custom modules.

Blazemeter permits us to have an alternate csv record every heap test motor. It must be done physically by duplicating the documents onto the Agent EC2 occurrences and have the same filename since the specialists allude to the Masters properties. Blazemeter permits us to parameterize the estimations of even filenames and have diverse csv records in every motor without offering us to the inconvenience of replicating documents into particular EC2 examples & holds the documents in a typical storehouse so it can be alluded from that point to every specialists.

BlazeMeter offers live observing of vital parameters of test servers when the test is running which empowers user to settle on the number & occasion sort for the test. It gives AWS Cloud watch integration. An account with IAM access must be made and AWS Access Key & Secret Key qualities must be designed so that the measurements are accessible in the Blazemeter s dashboard. This highlights helps us to see how the benefits in the cloud are responding to tests and help us likewise tune the base. While performing burden testing, it is essential not just to screen your Web Servers & Databases additionally the specialists from where the heap is produced . The New Relic plugin issues us the front end KPIs and back end KPIs. It s frontend KPIs give knowledge on what number of clients are really attempting to get to your site, versatile site or portable applications. It s backend KPIs demonstrate what number of clients are getting past to your applications.

3.5.2 Test result

Following table shows results evaluated after execution of 43 applications (20 java, 16 .net and 7 php based) for this tool: NOTE:values are considered for 2 decimal points only

Virtual User	Technology of Application	Response Time (sec)	Memory Utilization (MB)	Success Rate (%)	Error Rate
1 10 100	Java	2.59 22.02 190.95	4.03 23.75 207.59	100	0
1 10 100	.Net	0.2 2.95 26.58	10.92 132.72 832.6	89.4	10.6
1 10 100	php	1.69 30.49 200.47	3.62 482.02 1GB+	92	8

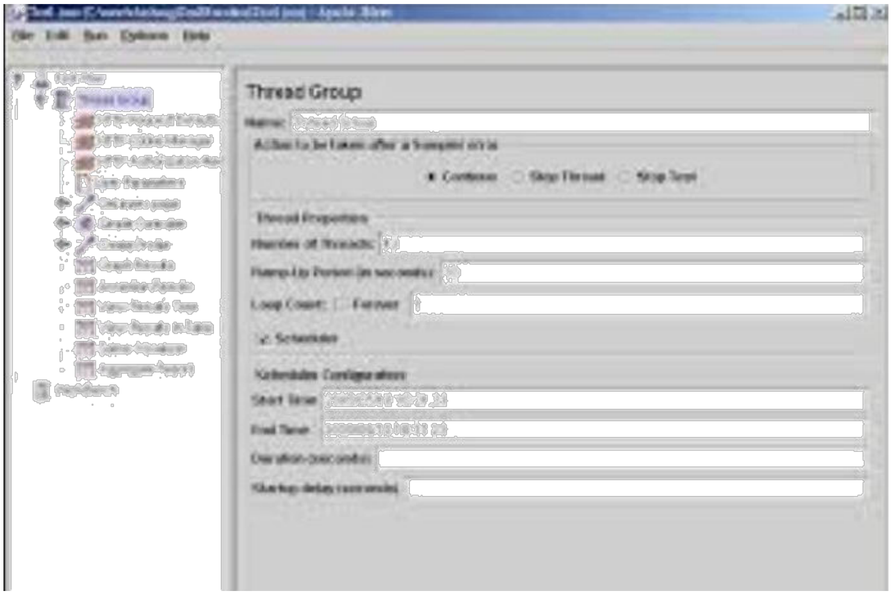
Table 3.5: Table

3.6 Comparison

Hence we deduced following comparison table form study of tools and results obtained from the same:

Name of tool	License type	Pros	Cons
LoadRunner	Commercial	Suitable outputs Easy to add plugins	Commercial software Installation hard
Jmeter	Open source	Open source Suitable outputs Installation easy Light weight tool Easy to add plugins	Missing features as compared to Load-Runner Improper GUI
locust	Open source	Open source Easy to add plugins	Linux based Unsuitable outputs
LoadTester	Freemium	Suitable outputs Only response time can be measured Installation easy	Need to pay for more features Useful for load test only
BlazeMeter	Commercial	Plugins can be added Installation easy	Commercial software Unsuitable outputs Improper GUI

Table 3.6: Table



Thread Group

Name:

Action to be taken after a Sampler error

● Continue ○ Stop Thread ○ Stop Test

Thread Properties

Number of Threads:

Ramp-Up Period (in seconds):

Loop Count: □ Forever

☑ Scheduler

Scheduler Configuration

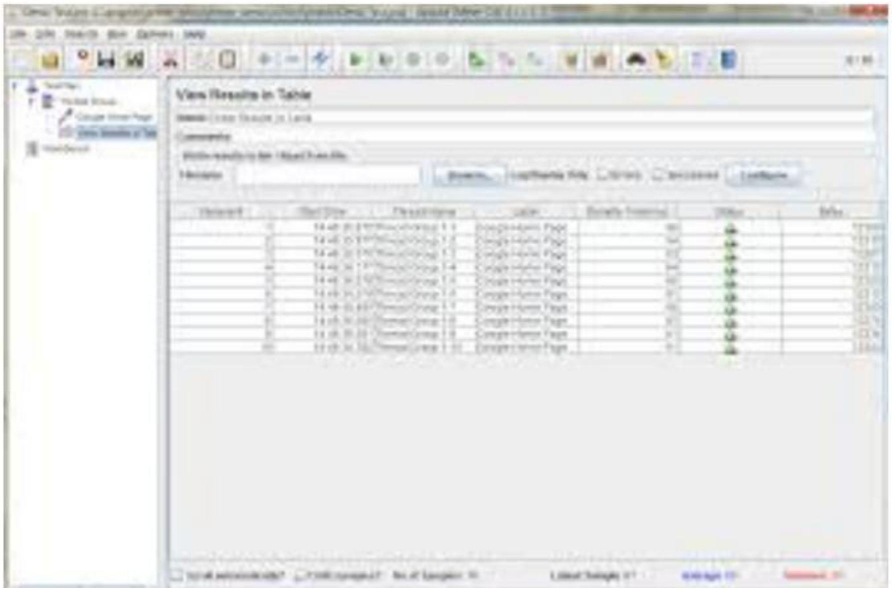
Start Time

End Time

Duration (seconds)

Startup delay (seconds)

Figure 3.10: JMeter Thread Setting



View Results in Table

Name:

Comments:

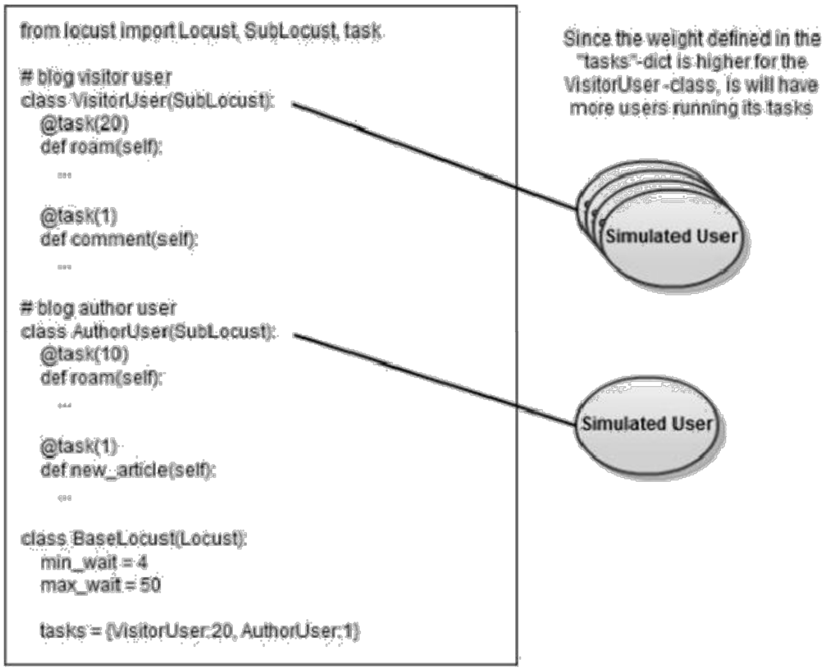
Filename

Browse...

Configure

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes
1	[illegible]	Thread Group 1-1	Google Home Page	[illegible]	✓	[illegible]
2	[illegible]	Thread Group 1-2	Google Home Page	[illegible]	✓	[illegible]
3	[illegible]	Thread Group 1-3	Google Home Page	[illegible]	✓	[illegible]
4	[illegible]	Thread Group 1-4	Google Home Page	[illegible]	✓	[illegible]
5	[illegible]	Thread Group 1-5	Google Home Page	[illegible]	✓	[illegible]
6	[illegible]	Thread Group 1-6	Google Home Page	[illegible]	✓	[illegible]
7	[illegible]	Thread Group 1-7	Google Home Page	[illegible]	✓	[illegible]
8	[illegible]	Thread Group 1-8	Google Home Page	[illegible]	✓	[illegible]
9	[illegible]	Thread Group 1-9	Google Home Page	[illegible]	✓	[illegible]
10	[illegible]	Thread Group 1-10	Google Home Page	[illegible]	✓	[illegible]

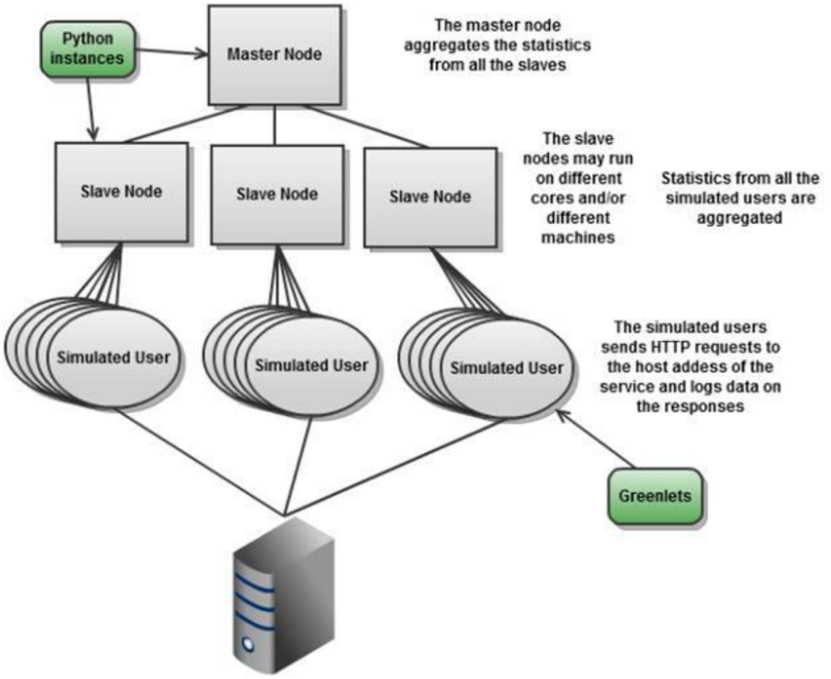
Figure 3.11: JMeter Working



```
from locust import Locust, SubLocust, task

# blog visitor user
class VisitorUser(SubLocust):
    @task(20)
    def roam(self):
        ...

    @task(1)
    def comment(self):
        ...

# blog author user
class AuthorUser(SubLocust):
    @task(10)
    def roam(self):
        ...

    @task(1)
    def new_article(self):
        ...

class BaseLocust(Locust):
    min_wait = 4
    max_wait = 50

    tasks = {VisitorUser:20, AuthorUser:1}
```

Since the weight defined in the "tasks"-dict is higher for the VisitorUser -class, is will have more users running its tasks

Figure 3.12: Locust Example



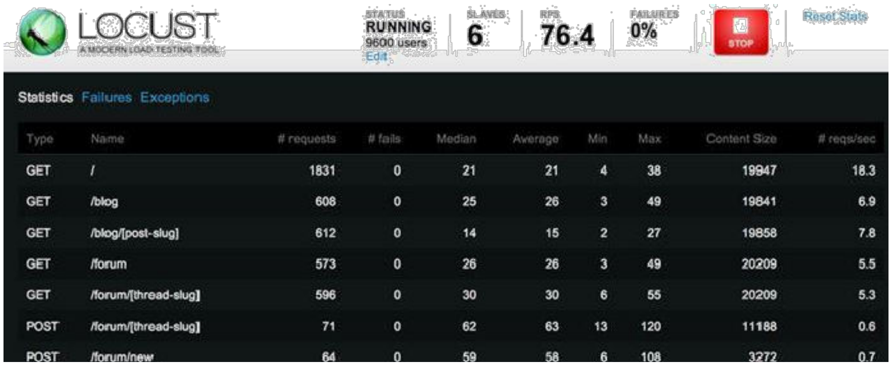
The master node aggregates the statistics from all the slaves

The slave nodes may run on different cores and/or different machines

Statistics from all the simulated users are aggregated

The simulated users sends HTTP requests to the host addess of the service and logs data on the responses

Figure 3.13: Locust Working Architecture



Type	Name	# requests	# fails	Median	Average	Min	Max	Content Size	# reqs/sec
GET	/	1831	0	21	21	4	38	19947	18.3
GET	/blog	608	0	25	26	3	49	19841	6.9
GET	/blog/[post-slug]	612	0	14	15	2	27	19858	7.8
GET	/forum	573	0	26	26	3	49	20209	5.5
GET	/forum/[thread-slug]	596	0	30	30	6	55	20209	5.3
POST	/forum/[thread-slug]	71	0	62	63	13	120	11188	0.6
POST	/forum/new	64	0	59	58	6	108	3272	0.7

Figure 3.14: Locust Running

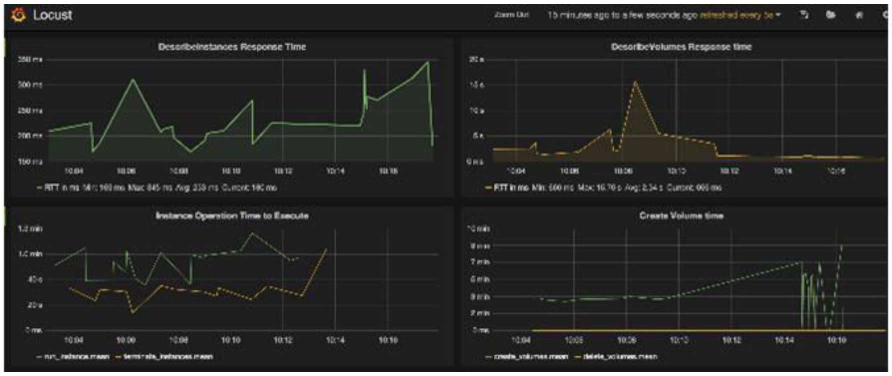


Figure 3.15: Locust Test Analysis

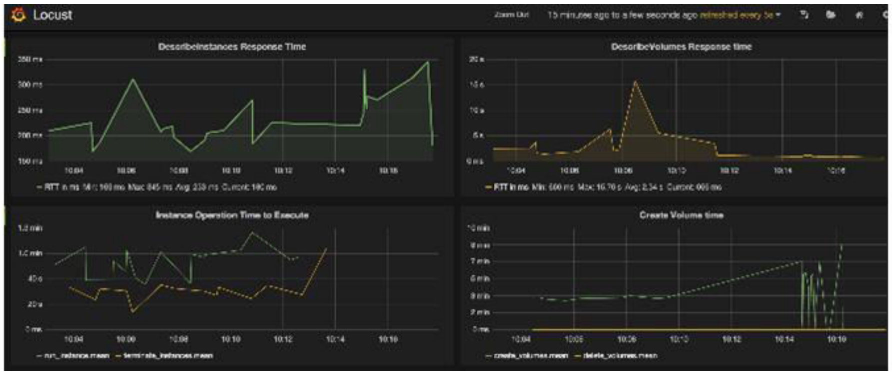


Figure 3.16: Locust Test Analysis

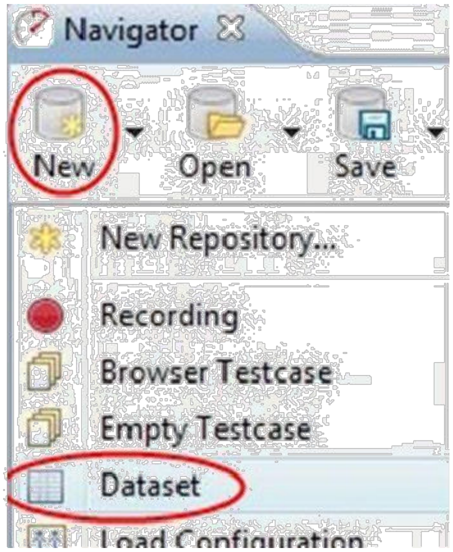
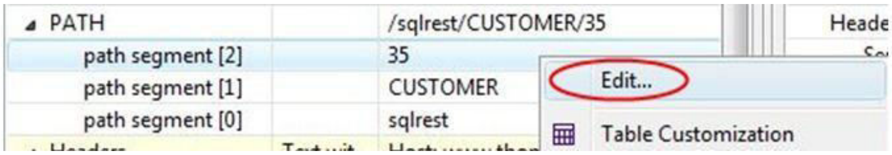


Figure 3.17: Load Test Creating Dataset



PATH		/sqlrest/CUSTOMER/35	Heade
path segment [2]		35	
path segment [1]		CUSTOMER	
path segment [0]		sqlrest	
Headers	Text wit	Host: www.than	

Figure 3.18: Loadtest Altering Datasource

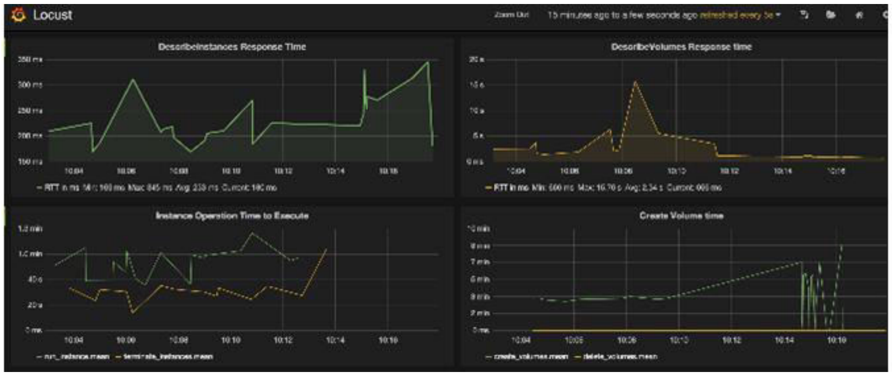

Locust
Zoom Out
15 minutes ago to a few seconds ago refreshed every 5s
DescribeInstances Response Time
350 ms
300 ms
250 ms
200 ms
150 ms
10:04
10:06
10:08
10:10
10:12
10:14
10:16
RTT in ms Min: 160 ms Max: 845 ms Avg: 233 ms Current: 180 ms
DescribeVolumes Response time
20 s
15 s
10 s
5 s
0 ms
RTT in ms Min: 600 ms Max: 16.70 s Avg: 2.34 s Current: 665 ms
Instance Operation Time to Execute
1.2 min
1.0 min
40 s
20 s
0 ms
run_instance.mean
terminate_instances.mean
Create Volume time
10 min
8 min
7 min
6 min
3 min
2 min
0 ms
create_volumes.mean
delete_volumes.mean

Figure 3.19: Locust Test Analysis

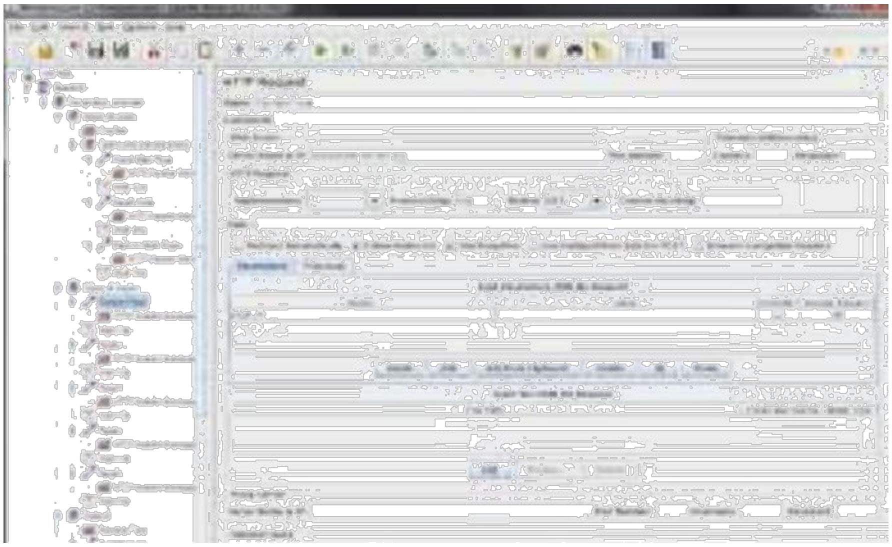


Figure 3.20: BlazeMeter Working

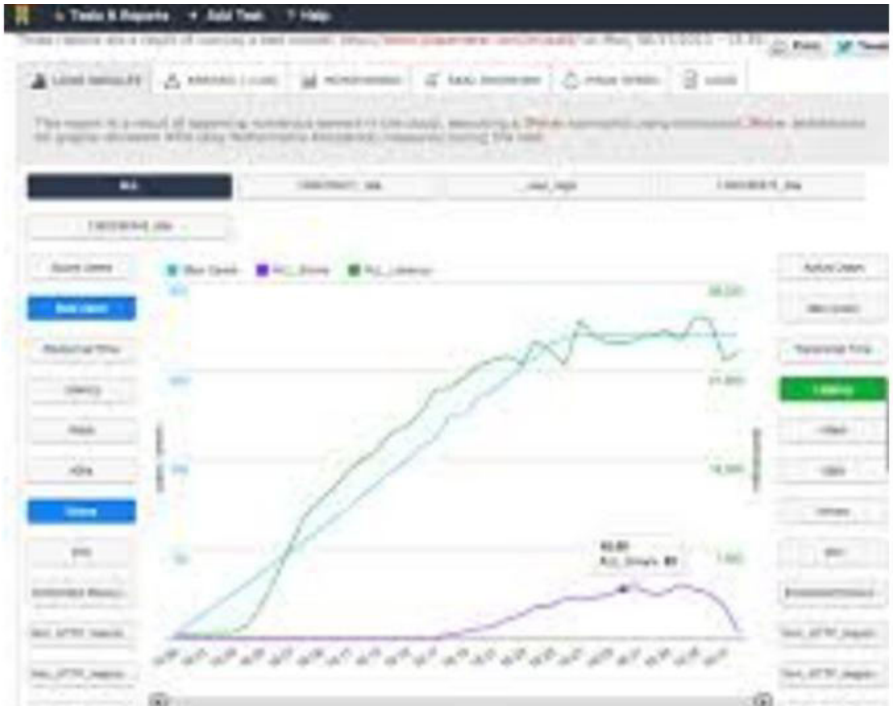


Figure 3.21: BlazeMeter Analysis

Chapter 4

Conclusions and future work

4.1 Conclusion

After detailed study of tools and their features, we found that these tools have some or other feature missing as compared to LoadRunner. However on close observation Jmeter was found to be useful as it has many advantages over others (Please refer section 3.6 for the same) Hence, to help organization achieve TMMI level 3, we conclude that HP LoadRunner tool should be replaced with Jmeter as it is proper, suitable and feasible option.

4.2 Future Work

Future work includes editing Jmeter as per organizations requirement and transferring of testing activities from LoadRunner to Jmeter.

References

[1] \Online resource: " http://en.wikipedia.org/wiki_Software_performance_testing"."

[2] \Online resource: " http://en.wikipedia.org/wiki/Performance_ engineering"."

[3] \Online resource: " http://en.wikipedia.org/wiki/Software_testing"."

[4] \Online resource: " http://www.st.com/web/en/"."

[5] \Online resource: " http://prateekvjoshi.com/2013/08/21/ why-do-we-need-performance-testing/"."

[6] K. Zhu, J. Fu, and Y. Li, \Research the performance testing and performance im-provement strategy in web application," in 2010 2nd International Conference on Education Technology and Computer, vol. 2, 2010.

[7] G. Jiang and S. Jiang, \A quick testing model of web performance based on testing flow and its application," in Web Information Systems and Applications Conference, 2009. WISA 2009. Sixth, pp. 57{61, IEEE, 2009.

[8] M. D. M. Suffian and F. R. Fahrurazi, \Performance testing: Analyzing differences of response time between performance testing tools," in Computer & Information Science (ICCIS), 2012 International Conference on, vol. 2, pp. 919{923, IEEE, 2012.

[9] M. Jovic and M. Hauswirth, \Performance testing of gui applications," in Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on, pp. 247{251, IEEE, 2010.

[10] S. Artzi, J. Dolby, S. H. Jensen, A. Moller, and F. Tip, \A framework for automated testing of javascript web applications," in Software Engineering (ICSE), 2011 33rd International Conference on, pp. 571{580, IEEE, 2011.

[11] O. Hamed and N. Kafri, \Performance testing for web based application architec-tures (. net vs. java ee)," in Networked Digital Technologies, 2009. NDT'09. First International Conference on, pp. 218{224, IEEE, 2009.

[12] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, \State of the art: Automated black-box web application vulnerability testing," in Security and Privacy (SP), 2010 IEEE Symposium on, pp. 332{345, IEEE, 2010.

[13] A. Freitas and R. Vieira, \An ontology for guiding performance testing," in Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web In-telligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01, pp. 400{407, IEEE Computer Society, 2014.

[14] \Online resource: " https://msdn.microsoft.com/en-us/library/bb924363. aspx"."

[15] \Online resource: " http://en.wikipedia.org/wiki/HP_LoadRunner"."

[16] \Online resource: " http://alternativeto.net/software/loadrunner/?platform=linux"."

[17] \Online resource: " http://www8.hp.com/us/en/software-solutions/ loadrunner-load-testing/index.html?jumpid=va_uwxy6ce9tr"."

[18] \Online resource: " http://www.wilsonmar.com/1loadrun.htm"."

[19] \Online resource: " https://msdn.microsoft.com/en-us/library/bb924356. aspx"."

[20] \Online resource: " https://wiki.apache.org/jmeter/JMeterArchitecturalOverview"."

[21] \Online resource: " http://blazemeter.com/blog/ jmeter-viable-open-source-alternative-loadrunner"."

[22] \Online resource: " http://jmeter.apache.org/"."

[23] \Online resource: " http://www.softwaretestingclub.com/forum/topics/ jmeter-vs-load-runner-performance-tool"."

[24] \Online resource: " http://en.wikipedia.org/wiki/Apache_JMeter"."

[25] M. R. Dhote and G. Sarate, \Performance testing complexity analysis on ajax-based web applications," Software, IEEE, vol. 30, no. 6, pp. 70{74, 2013.

[26] \Online resource: " http://killera.github.io/test/2013/07/29/Comparison_between_JMeter_and_Locust/"."

[27] A. I. Wasserman, \Software engineering issues for mobile application development," in Proceedings of the FSE/SDP workshop on Future of software engineering research, pp. 397{400, ACM, 2010.

[28] M. A. S. Netto, S. Menon, H. V. Vieira, L. T. Costa, F. M. de Oliveira, R. Saad, and A. Zorzo, \Evaluating load generation in virtualized environments for software performance testing," in Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, pp. 993{1000, IEEE, 2011.

[29] G.-h. Kim, H.-c. Moon, G.-P. Song, and S.-K. Shin, \Software performance testing scheme using virtualization technology," in Ubiquitous Information Technologies & Applications, 2009. ICUT'09. Proceedings of the 4th International Conference on, pp. 1{5, IEEE, 2009.

[30] \Online resource: " http://uu.diva-portal.org/smash/get/diva2:685934/FULLTEXT01.pdf"."

[31] \Online resource: " http://www.webperformance.com/load-testing/blog/2015/04/api-load-testing/"."

[32] \Online resource: " https://docs.blazemeter.com/customer/portal/articles/1808038-blazemeter-rest-apis"."

[33] R. Mansharamani, A. Khanapurkar, B. Mathew, and R. Subramanyan, \Performance testing: Far from steady state," in Computer Software and Applications Con-ference Workshops (COMPSACW), 2010 IEEE 34th Annual, pp. 341{346, IEEE, 2010.

[34] X. Che and S. Maag, \Passive testing on performance requirements of network proto-cols," in Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on, pp. 1439{1444, IEEE, 2013.

[35] Q. Wu and Y. Wang, \Performance testing and optimization of j2ee-based web applications," in Education Technology and Computer Science (ETCS), 2010 Second International Workshop on, vol. 2, pp. 681{683, IEEE, 2010.