Integrated Tool for Web Application Performance Evaluation

S. K. Wanniarachchi, U. V. Wanniarachchi, K. H. Samarappuli, V. Nanayakkara and S. Sooriyarachchi
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka

Abstract— **Performance Evaluation is essential to ensure that the web applications are providing a satisfactory level of service to its users in terms of scalability, stability, and throughput or response time. Web application performance evaluation process includes experimental designing, generating workloads for experiments, executing test while monitoring resource utilization of server and client, and representing results after test execution. This paper analyzes a number of popular performance evaluation tools in terms of how well they contribute to this complete process of evaluation. The results show that majority of the existing tools for web application performance evaluation have several limitations as far as the entire testing process is concerned, for example experimental design and resource monitoring do not co-exist in a single tool. Our contribution is to develop a fully fledged, open source tool which we name as WingPerf. It facilitates the entire web application performance evaluation process. This paper also presents the architecture and the functionalities of WingPerf.**

Keywords – **Web application, Performance evaluation, Open source, Experimental design, Workload generation, Resource monitoring**

I. Introduction

Web application performance evaluation is a broad domain of depiction, due to their exposure to large number of end users. Performance means, identifying, to which degree the web application meets stipulated goals that assures user satisfaction. It is required to measure the capacity of a web application in its domain, to verify that it can meet performance objectives without failures, when deployed in the real world.

Let us consider an e-learning application hosted on a server. It is intended to provide the service to a variety of clients (users) simultaneously or consecutively. Hence, range of tolerable load by that application should be premeditated in order to offer an accurate and continuous service. Forexample, consider 100 students doing an on-line quiz. All 100 students should be able to access the system and do the quiz simultaneously; system response to the users also should be accurate and compatible with user expectations. Furthermore, the system should respond to all users in similar manner. User satisfaction may degrade, if the system is vulnerable to any failure(s). Performance evaluation for web application is critical at this point to assure that a web application caters its user expectations. Even though there are wide variety of open source and proprietary tool(s)/framework(s) [1], [2], [3] that can be used for web application performance evaluation, the deficiencies in those tools motivated us to design and implement a new open source integrated tool, WingPerf.

This paper covers two objectives, one is to identify and analyze existing tools and frameworks that facilitate web application performance evaluation, including their capabilities and limitations. Second is to introduce WingPerf as a solution to the discussed problem that eliminates the shortcomings compared to other tools and frameworks we analyzed.

The remainder of this paper is structured as follows. Section 2 discusses the problem in detail and our proposed solution. In section 3, it describes the related work, with the evaluation criteria defined for web application performance evaluation tool(s)/framework(s). Section 4 includes complete comparison between existing tools and frameworks and a tool analysis according to the defined criteria in the previous section. Section 5 introduces WingPerf as the solution to the discussed problem and defends the claims with relevant facts and graphs. Finally, Section 6 concludes the discussion with possible future improvements.

II. Problem Statement

There are numerous case studies that evaluate web application performance [4], [5]. The key purpose of web application performance evaluation is to determine the throughput, responsiveness, availability and reliability of a web application under a given workload and assist benchmarking the application. Furthermore, identifying the web application/server bottleneck(s) is also expected from a performance evaluation. Performance forecasting and performance tuning are other pros in web application performance evaluation [6].

In addition to that, conducting a performance evaluation helps to determine the system's hardware configuration required to run that web application in real environment. Hence, Performance evaluation of a web application is extremely important before launching it live to determine whether the user expectations have been achieved.

The important aspects need to be considered in performance evaluation of web applications are,

1) Defining performance and workload parameters
2) Experimental design
3) Workload generation and characterization
4) Test execution and resource monitoring
5) Results analyzing and representation

A. Defining performance and workload parameters

This is the initial step in performance evaluation of a web application. Performance analyst defines parameters needed for workload generation and test execution depending on the evaluation type need to be conducted.

B. *Experimental design*

The main purpose of experimental design is to obtain maximum information with minimum experiments. The cost of performance evaluation increases when the number of experiments increases [7]. There are numerous varieties of experimental designs [8] that lead to conduct an effective performance evaluation against web applications. Three most frequently used designs are simple, full factorial and fractional factorial design [8].

C. *Workload generation and characterization*

Workload is the most crucial part of any performance evaluation. If the workload is not correctly selected, the evaluation may lead to wrong conclusions. The concept of workload characterization defines to observe the real user environment with its characteristic, and develop a workload model that can be used repeatedly [8]. Workload generation can be categorized in to real workload and synthetic workload. Real workload is the workload that can be observed live in operation. But it is not feasible to apply a real workload to perform a test under controlled environment. Therefore a mimic of real workload or a synthetic workload is used. A synthetic workload is said to be representative of a real workload if both workloads result in similar performance when submitted to a system [9]. Synthetic workload can be classified in to either a trace based workload or a workload generators [10]. Trace based workloads are the cached workloads extracted from server logs, and workload generators allows to generate both request and session oriented workloads.

D. *Test execution and resource monitoring*

The process of test execution depends on features such as test execution tool and testing environment. A web application performance test execution involves defining the testing environment, running the test, monitoring data, monitoring resource utilization, archiving obtained results and data logging. Quantitative measurements such as response time, reply rate, connection rate can be achieved from those tests and qualitative measurements such as reliability, scalability, interoperability can also be evaluated. An effective test execution plan can be accomplished by applying experimental design and test replication.

E. *Results analyzing and representation*

It is important to identify results and outcomes of the experiments. The performance evaluation results can be interpreted in a meaningful manner by using graphical charts and histograms. Result representation should be done in an easy-to-understand manner.

A proper tool, which addresses all the aspects of above described process, is needed in order to conduct an effective test and provide constructive results. Such a tool needs to be flexible and user friendly with the independency for the performance analyst to conduct any type of evaluation. Even though there exist numerous tools that can support this problem, limitations of them in the entire testing process obstruct the analyst to carry out a successful and acceptable performance evaluation. Characteristics of such tools are compared and contrast in the related work section.

III. RELATED WORK

There are several methodologies to measure web application performance. Performance Testing Guidance for Web Applications [6] provides an intuitive approach including seven core activities to perform a test, specifically for web applications. A systematic approach for performance evaluation is represented in [8] including 10 steps. It describes that most of the performance problems are unique [8]; hence the performance metrics, workloads and testing techniques cannot be used for all types of test cases. However, there are steps that are common to all performance evaluation scenarios. Although these steps explain performance evaluation process in general, they can be used in web application performance evaluation as well.

Table 1 describes the criteria used to evaluate the exiting tools/frameworks, which includes 9 features. Experimental design feature determines the level of flexibility provided by a tool in designing test sequences. We have considered the ability to do simple test design, full factorial test design and $2^{(k-r)}$ fractional factorial test designs as features that should be facilitated in a test design tool. The two categories of workload types are real and synthetic [8].
The ability to generate synthetic workloads is a major concern when selecting a workload generation tool. Simulation is based on recorded real user behavior and then varies parameters in order to create behavior explicitly richer and realistic. User emulation referred to as the reproduction of simulated real user behavior, which is another aspect we would consider in a workload generation tool. In general, test execution tool comprises with workload generation. Hence, the main requirements of a test execution tool are workload generation, execute the designed test and provide the output result. Resource utilization of both web server and client machines should be monitored while the test is executing. In the field of monitoring, granularity of the monitored data is the fundamental feature. A monitoring tool should support finer data granularity, since some tests may remain only for a few seconds. Nevertheless, some monitoring tools may function for both server and client monitoring with coarse data which is not in accordance with our requirement. The results should be archived for analyzing purposes, after the test execution is finished. Capability of representing results in comprehensive manner, such as reports, customized graphs or tableau format is essential in analyzing data and facilitating recommendations. Data should be archived and extracted for analyzing effortlessly. Having such features in a test execution and a monitoring tool is highly required.

Recommendations are referred to as providing appropriate suggestion on capacity of the web application which was under the accomplished test. Recommendations can be resolved by analyzing input with relevant output or by any other statistical analysis. In addition to these features, any tool that featured above aspects should be effortless to use with proper user interfaces.

TABLE I
EVALUATION CRITERIA

Feature/capability	Fully featured	Partially featured	None
Experimental design	Facilitate simple test design, full factorial test design and $2^{(k-r)}$ fractional factorial test design	Either one of the test design types described in fully features section or test replications or test with ramping [19]	Not supporting any kind of test design
Workload types	Generating synthetic workload		No workload generation
User emulation	Simulate one user and reproduce that user for required number		No user emulation
Test execution	Execute the designed test		No test execution
Resource monitoring	Client and server monitoring , Fine data granularity	Either client or server monitoring coarse data granularity	No resource monitoring
Results Representation	Archive test results Result extraction	Results extraction	Do not have this feature
Analyzing test results and facilitate benchmarking/recommendations	Plotting customized graphs	Do not plot graphs Results in tableau format	No graphs or results in tableau format
Ease of use	Fine user interfaces No need of any programming knowledge to use the tool	User interfaces But need some programming knowledge to use	Command line tools
Open source	Product is freely available		Product is proprietary

Performance analyst should be able to use such tool without any need of a programming skill. If the tool is open source it is considered as a positive feature for tool evaluation. Table 1 summarizes the desired features in a performance evaluation tool discussed above. Table 1 also shows the criteria that measures the level of availability of each feature in a tool.

Table 2 analyzes the main features of existing tools/frameworks that support in the process of web application performance evaluation according to the evaluation criteria in Table 1. Each of these tools can be used in at least one step in web application performance evaluation process discussed in section (II).

DOE++ [11] is a tool which is specially developed for experimental designing purposes. DOE++ provides a large variety of experimental design types, detailed analysis of experimental data and extensive plotting capabilities to present the results graphically [12]. However, DOE++ does not have the capability of conducting a complete web application performance test as shown in Table 2. Furthermore, DOE++ is a proprietary tool which works only on windows environment.

TestCaseGenerator is a tool which could be used for test generation in combinatorial test scenarios and it provides the facility to manually edit the generated test case [13]. This is an open source tool, which comes as a .net project, working only on windows environment.

Apache Jmeter is a powerful, easy-to-use, open source tool which was originally designed for testing web applications [14]. According to Table 2, Jmeter is providing limited test designing facility, simulating heavy loads in the server, execute the test and get the result in a table view or tree view. It does not provide any graphical representation of results or any resource monitoring facility.

Webserver Stress tool [15] is a robust, flexible and powerful HTTP – client/server test application, which identifies and measure performance issues of web applications. It allows generating synthetic workload on the server, user emulation, test execution and providing fine representation of results after test is finished. When analyzing this tool under our evaluation criteria, we found that it lacks experimental design capability and the transparency of experimental designing (See Table 2). Also it only provides the client resource monitoring facility and does not monitor the server resources during the test execution. Webserver Stress tool is easy to use tool which is proprietary and works on windows environment.

WAPT [16] is another proprietary, easy to use tool, working on windows environment, which is for evaluating web applications performance. WAPT provides limited test designing facility depending on the test type. It provides the facility to emulate the real user and generate a workload identical to the real workload on the server and execute the test. Test results are provided in an informative way and represented as descriptive graphs and reports. WAPT only monitors the client resource and do not monitor the server resources.

TABLE I
FEATURE OF CURRENT TOOL(S)/ FRAMEWORK(S) IN THE PROCESS OF WEB APPLICATION PERFORMANCE EVALUATION

Feature/capability	D OE ++	TestCaseGenerator	Apache Jmeter	Webserver Stress Tool	Wapt	Webload	SURGE	SPECweb99	TPC-w	Mercury LoadRunner	WebSTONE	Httperf	Autobench	RRDTool	MRTG
Experimental design	●	◌	○	◌	◌	○	○	○	○	◌	○	○	◌	○	○
Workload types	○	○	●	●	●	●	●	●	●	●	◍	◍	●	○	○
User emulation	○	○	●	●	●	●	●	●	●	●	◍	◍	●	○	○
Test execution	○	○	●	●	●	●	●	●	●	●	◍	◍	●	○	○
Results Representation	○	○	●	●	●	●	○	●	○	●	○	◍	●	◍	●
Analyzing test results and Facilitate Recommendations for benchmarking	●	○	●	●	●	●	○	○	○	●	○	○	○	◍	●
Resource monitoring	○	○	○	◌	◌	○	○	○	○	◌	◌	◌	◌	○	◌
Ease of use	●	●	●	●	●	◌	○	○	○	●	○	○	○	○	○
Open source	○	●	●	○	○	●	●	○	○	○	◍	◍	●	◍	●

WebLOAD [17] is an easy to use, open source product which can be used in web application performance evaluation process. It has almost all the features we consider in the tool evaluation criteria as shown in Table 2. However, it lacks test designing ability and it is not easy to use though it comes with fine user interfaces.

Mercury LoadRunner [18] is an easy to use, proprietary tool, working on windows environment, which can be used for performance testing to determine scalability, behavior and performance of the application. It supports most of the steps we consider in web application performance evaluation process, with less experimental designing facility and resource monitoring facility as in Table 2.

Httperf [2] is open source, command line tool for workload generation and test execution tool for web application performance testing. This tool does not have some features we consider in our tool evaluation criteria as this is a command line tool. However it has a large flexibility in workload generation. This tool does not provide any experimental designing facility. Results representation is done in command line and if the user wanted, the result can be saved to a text file. User should format these results before analyzing, because the way the results are given by the tool is not in easy to be analyzed. Furthermore httperf monitors the client resource while the test is executing.

Autobench [19] is an extended version of httperf which has almost all the features of httperf. It allows replicating experiments with a ramp.

MRTG [20] is a server resource monitoring tool which can be used to monitor server resources. MRTG provides monitoring results with fewer granularities and it is not easy to archive results. RRDTool [21] is a data logging tool and it can be used in the resource monitoring stage in the web application performance evaluation process along with SNMP. RRDTool can provide the monitoring result in finer data granularity and it is easy to archive results than MRTG tool.

IV. WINGPERF

Our solution, WingPerf is an open source, user friendly, comprehensive tool for the purpose of designing and executing tests to analyze web application performance. WingPerf integrates several existing open source tools and libraries related to web application performance evaluation, where other components are built from scratch. The target of all these components is to allow performance analyst to create test scenarios on top of the application, execute tests and analyze results. WingPerf gains its comprehensiveness by providing all broad aspects related to web application performance evaluation, discussed in evaluation criteria in Section (III) as shown in the last column in Table 2. One of the key features in WingPerf is the higher flexibility in experimental designing. WingPerf gives the freedom for the performance analyst to design performance tests according to his/her preference. WingPerf provides the facility to design a one factor test design or full factorial design with any number of factors and levels. It provides fractional factorial test design with any number of factors with two levels. Therefore it is straight forward for the performance analyst to develop the desired performance test cases. It also allows replication of tests which are not using any experimental design method. Since workload is the most crucial input from all above, WingPerf offers synthetic workload options classified into session based, request based, log based etc. WingPerf provides the facility to record a user session using a URL recorder.

WingPerf uses httperf, for workload generation and test execution. Httperf provides flexibility for generating various HTTP workloads and for measuring server/application performance [2].

WingPerf provides for monitoring server resource utilizations while the test is in progress. For an example assume 100 students are doing an on-line quiz in an e-learning application simultaneously. WingPerf can emulate this situation by generating a synthetic workload and measure performance of the application with web server's resource utilization such as CPU, memory and disk. The server resource utilization should be remotely monitored. A SNMP [22] daemon such as NET-SNMP [23] should be installed and run in the server machine where SNMP utility library should be installed in the client machine where WingPerf is running. WingPerf uses RRDtool [2] for logging data obtained from SNMP walks. RRDtool is configured to log data during a period of a few seconds. Furthermore, WingPerf has the ability to run an evaluation without monitoring or carry out a real monitoring session for the web server. This scenario implies the versatility and the level of independence of WingPerf. After the evaluation is done, httperf results are shown in tabular form for each test case and the evaluation log is shown for the performance analyst for analyzing purposes.

Analysis of test results is the most crucial step because it governs the decisions regarding the web application. WingPerf facilitates analysts to analyze performance by plotting graphs and evaluate the behavior of the outputs against the inputs. Area, bar, scatter, lines are the plot types available in reporting with the total independence for the performance analyst to plot using most of the results of httperf. We have used CPAN Perl modules like GD::Graph [24] as the graph generator. To make a decision regarding the performance of a web application, one test procedure is insufficient. Considering the previous evaluation results, performance analyst can re-evaluate the web application and conclude performance evaluation to deliver proper decisions or benchmark the web application for the authorized parties. Following section describes a case study on effective usage of WingPerf in real applications.

V. WINGPERF CASE STUDY

A. *Case Study 1*

Consider a situation of 50 students, access a learning management system, log in, do a quiz (10 minutes) and then log out. Analyst is able to measure the performance of the application and other performance statistics at this situation.

Among the workload generation mechanism provided in WingPerf, session recording is used to record the real student behavior on the application under real environment conditions with object characteristics (e.g.: content size) and user characteristics (e.g.: user think time). Recorded file includes a sequence of URI's, request method, think-time and burst length parameters as shown in Fig 1.

```
/moodle method=GET think=1
    /moodle/ method=GET
    /moodle/theme/standardwhite/styles.php
method=GET
    /moodle/lib/javascript-mod.php method=GET
    /moodle/calendar/overlib.cfg.php method=GET
    /moodle/login/index.php method=GET
    /moodle/theme/standardwhite/styles.php
method=GET
/safebrowsing/downloads?client=Firefox&appver=3.0.8&
pver=2.2&wrkey=AKEgNitdkQXnKGXHF_
-
-
-

name=\"questionids\"\<CR>\<CR>1,2,3,4,5,6,7,8,10,9\<
CR>-----------------------------
20981443291959896731144994537--\<CR>" think=447
    /moodle/mod/quiz/review.php?attempt=16
method=GET
    /moodle/theme/standardwhite/styles.php
method=GET
    /moodle/login/logout.php?sesskey=c72MXRi7Jm
method=GET
    /moodle/ method=GET
    /moodle/theme/standardwhite/styles.php
method=GET
    /moodle/calendar/overlib.cfg.php method=GET
```
Fig 1: Recorded url file

Fig 2 through Fig 5 shows the variation of memory and CPU utilization of the server during the execution of the said workload.

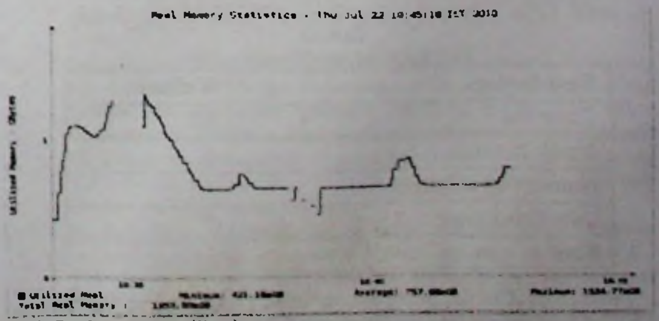

Fig 2: Real memory utilization

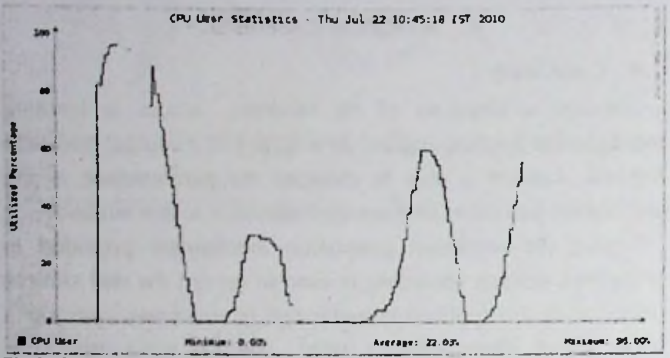

Fig 3: CPU User utilization of the server while test is in progress

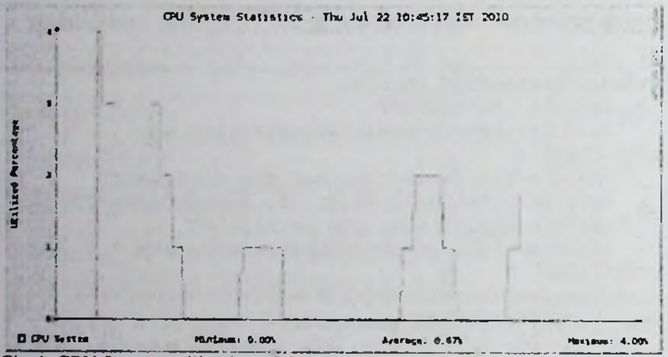

Fig 4: CPU System utilization of the server while test is in progress

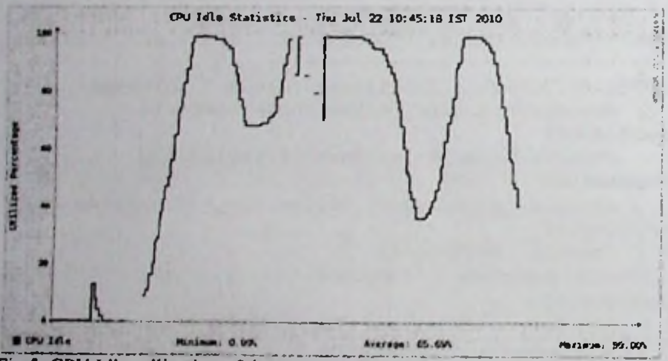

Fig 5: CPU Idle utilization of the server while test is in progress

After user emulation, it was simulated to represent 50 students and was executed on the Moodle application. Factors and the selected levels for the experiment are given in Table 3.

TABLE3
TEST FACTORS AND THEIR VALUES

Test factors	Values
Time out	60
Think timeout	5
Number of sessions	50
User-think time	1
Rate	5

Table 4 summarizes the test results whereas Table 5 summarizes the client side resource utilization.

TABLE 4
SUMMARY OF TEST RESULTS

Test No: 1	
Avg Conns Time(ms)	2594.6
Avg Rep Rate(replies/s)	23
Conn Length(rep/conn)	1
Connection Rate(conns/s)	22.9
Request Rate(req/s)	220.3
Test Duration(sec)	159.312
Total Connections	3650
Total Errors	3550
Total Replies	3650
Total Requests	35100

TABLE 5
CLIENT RESOURCE UTILIZATION

User CPU Time(s)	8.97
User CPU Time %	5.60%
System CPU Time(s)	118.64
System CPU Time %	74.50%
Total CPU Time %	80.10%

B. *Case study 2*

This section describes performance evaluation of actually hosted application. It is recommended to test the performance in a separate testing environment to avoid the network bottleneck. However, WingPerf facilitates performance evaluation in any environment.

Consider a situation that arises frequently, such as a user accessing online news reporting application and reading through news.

```
/ method=GET think=1
/2010/07/22/main_News.asp method=GET think=2
/2010/07/22/b00001.css method=GET think=2
    /2010/07/22/main_News.asp method=GET
-
-
-
    /images/extra/suo_mps-lo.jpg methcd=GET
    /images/extra/suo-classifieds-lo.jpg method=GET
    /images/extra/govt_gazette-lo.jpg method=GET
    /2010/07/22/z_bus350.jpg method=GET
    /images/extra/print_icon2.gif method=GET
    /images/extra/print_icon1.gif method=GET
/x/216283306/false/p_1041991639=0 method=GET think=3
    /p method=GET
```
Fig 5: Real user behavior on the application

This case study simulates the user session (in Fig 5) to represent 5 users accessing the same application at a rate of 1 session per second and executes the test. Table 6 shows the selected factors and their values.

TABLE 6
TEST FACTORS AND THEIR VALUES

Test factors	Values
Time out	30
Think timeout	5
Number of sessions	5
User-think time	1
Rate	1

TABLE 7
SUMMARY OF TEST RESULTS

Test No: 1	
Avg Conns Time(ms)	25004.7
Avg Rep Rate(replies/s)	8.3
Conn Length(rep/conn)	17
Connection Rate(conns/s)	0.5
Request Rate(req/s)	7.9
Test Duration(sec)	43.099
Total Connections	20
Total Errors	0
Total Replies	340
Total Requests	340

TABLE 8
CLIENT RESOURCE UTILIZATION

User CPU Time %	9.70%
User CPU Time(s)	4.18
System CPU Time %	88.20%
System CPU Time(s)	38.01
Total CPU Time %	97.90%

Table 7 summarizes the test results while Table 8 summarizes the client side resource utilizations.

VI. CONCLUSION

Companies, institutes and other organizations increasingly rely on websites to conduct their businesses and other tasks to maintain competitive advantage. Yet most lack insight into what users are experiencing on their website and have no means for identifying the issues that drive users away. In this paper, we focused on estimation of web application performance to ensure that the acceptable services are provided by that application in terms of user expectations and assist benchmarking the application.

We have Analyzed characteristics of several existing tools/ frameworks including their capabilities and limitations focusing on the entire testing process. Since they are focused on segregated tasks none is having the features that are required in accomplishing an accurate performance test against a web application including experimental designing, workload generation, test execution while monitoring client and server, results representation with analysis. Our contribution is to introduce a solution that can estimate performance of web applications, WingPerf. It is an open source fully fledged integrated tool that is capable of facilitating performance evaluation of web applications and assist analyst to benchmark it together with all the aspects of test process. WingPerf addresses the important aspects that need to be considered in performance evaluation of web applications; Defining performance and workload parameters, Experimental design, Workload generation and characterization, Test Execution and Results representation with analyzing. WingPerf is flexible compared to the current tools that we have described and compared in the related work section.

WingPerf is capable of performance evaluation with the assumption that system under test is isolated from the network; hence it is with the potential to advance to monitor the network behavior as well. As described in the section 4, experimental designing in fractional factorial design is modeled only for 2 levels of factors which can be increased to any number of levels according to user requirements. Improved granularity on monitoring can be obtained to provide more precise results against performance tests.

REFERENCES

[1] P. Barford and M. Crovella, "Generating representative Web workloads for network and server performance evaluation," *SIGMETRICS Perform. Eval. Rev.*, 26(1), 1998, pp. 151-160.

[2] D. Mosberger and T. Jin, "httperf&mdash;a tool for measuring web server performance," *SIGMETRICS Perform. Eval. Rev.*, 26(3), 1998, pp. 31-37.

[3] D. Garcia, "TPC-W E-Commerce Benchmark Evaluation," *COMPUTER -IEEE COMPUTER SOCIETY-*, 36, 2003, pp. 42-49.

[4] A. Kamra, V. Misra, and E. Nahum, "Yaksha: a self-tuning controller for managing the performance of 3-tiered web sites," *Twelfth IEEE International Workshop on Quality of Service 2004. IWQOS 2004.*, Montreal, QC, Canada , pp. 47-56.

[5] L. Titchkosky, M. Arlitt, and C. Williamson, "A performance comparison of dynamic Web technologies," *SIGMETRICS Perform. Eval. Rev.*, 31(3), 2003, pp. 2-11.

[6] J. Meier and Books24x7, Inc., *Performance testing guidance for web applications patterns & practices*, [Redmond, Wash.] Microsoft Press, 2007.

[7] D. Hoskins, R.C. Turban, and C.J. Colbourn, "Experimental designs in software engineering: d-optimal designs and covering arrays," *Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research*, Newport Beach, CA, USA: ACM, 2004, pp. 55-66.

[8] Raj Jain, *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling*, Canada: Wiley Computer Publishing, John Wiley & Sons, Inc, 1991.

[9] M. Shams, D. Krishnamurthy, and B. Far, "A model-based approach for testing the performance of web applications," *Proceedings of the 3rd international workshop on Software quality assurance*, Portland, Oregon: ACM, 2006, pp. 54-61.

[10] R. Peña-Ortiz, J. Sahuquillo, A. Pont, and J.A. Gil, "Modeling continuous changes of the user's dynamic behavior in the WWW," *Proceedings of the 5th international workshop on Software and performance*, Palma, Illes Balears, Spain: ACM, 2005, pp. 175-180.

[11] "Design of Experiments Software (DOE Software Tool) - ReliaSoft DOE++ for Experiment Design and Analysis."

[12] "Experimental Design Software: ReliaSoft's DOE++ for Experiment Design and Analysis."

[13] M.Bulmahn, *TestCaseGenerator A Quick Start Tutorial*.

[14] E. Halili and Books24x7, Inc., *Apache JMeter a practical beginner's guide to automated testing and performance measurement for your websites*, Birmingham, U.K., Packt Publishing Ltd., 2008.
[15] Paessler Software Solutions., *Webserver stress tool*, Germany, Paessler Software Solutions, 2002.
[16] "WAPT - Web Application Load, Stress and Performance Testing."
[17] RadView Software, *WebLOAD Quick Start*.
[18] "Performance testing and LoadRunner.".[Online].Available: http://www.webdotdev.com/nvd/content/view/977/. [Accessed: Oct. 2, 2009]
[19] "Autobench.".[Online].Available: http://www.xenoclast.org/autobench/. [Accessed: Sep. 20, 2009]
[20] Tobi Oetiker. (January, 2006). "MRTG Documentation". [Online]. Available: http://oss.oetiker.ch/mrtg/doc/index.en.html. [Accessed: Oct. 28, 2009]
[21] T. Oetiker, "RRDtool - RRDtool Documentation". [Online]. Available: http://oss.oetiker.ch/rrdtool/doc/index.en.html. [Accessed: Nov. 15, 2009]
[22] "RFC 1157 (rfc1157) - Simple Network Management Protocol (SNMP) ". [Online].Available: http://www.faqs.org/rfcs/rfc1157.html
[23] "Net-SNMP.".[Online].Available:http://www.net-snmp.org/docs/readmefiles.html. [Accessed: Sep. 7, 2009]
[24] "GD::Graph - search.cpan.org," *CPAN*. [Online].Available: http://search.cpan.org/~bwarfield/GDGraph-1.44/Graph.pm. [Accessed: Dec. 4, 2009]