Scenario-Based Performance Analysis Of Web-Based Systems

R. Srinivasa Perumal
Department of Computer Science Pondicherry University, Pondicherry, India.
sriperl_saro@yahoo.com

P. Dhavachelvan
Department of Computer Science, Pondicherry University, Pondicherry, India.
pd_chelvoume@yahoo.co.in

ABSTRACT

In the real world distributed system development scenario, implementation and testing a prototype distributed system is essential and its effectiveness is proved in various research works. This is highly facilitated in web based software application developments where the complexity is very high and the rates are undefined. The workload analysis in web based applications play a vital role for scheduling the request and responses. As the quality of scheduling depends on the operating point of the web application, a representative workload model is significant for fixing the region of safety operations. In this perspective, a hierarchical model is proposed to ease the mixing of performance analysis into the system design process through the scenario-driven analysis method. This unique model uses variety of statistical tools to validate the performance of the building software under various loading conditions. This tool based analogy evaluates the systems parameters like sessions, hits and pages on a single target-platform component. These analyzes performed for different user defined scenarios as like ramp-up, fixed and periodic and promisable results are obtained to guide the software development . A well defined rule based system provides a prototype decision making tool for providing guidelines to identify the safety operation regions of a web application domain.

Categories and Subject Descriptors:

D.2.5 [Testing and Debugging]: Testing tool, H.3.5 [Information storage and retrieval]: Web-based services, G.3. [Probability and Statistics]: Statistical Computing.

General Terms

Verification, Performance.

Keywords

Web service testing, Distributed system, Stress Testing.

1 INTRODUCTION

Validation and verification are the two major activities in developing any application system [5].The motivation of testing the system is improve the performance and quality of the system.

Software quality mainly focuses to decrease the failure rate of the system .But up to date very limited tools are available for automating the whole testing process and to achieve the acceptable solution.Quality of the software assured by means of testing, since it verifies the actual behavior of a program. Software developers and testers use several testing techniques to ensure their software has various qualities [1] [3] [4].

Testing activities support quality assurance by executing the software being calculated to provide the confirmation on the actual program behavior [8]. The vital goal of software testing is to help the developers in developing systems with high quality. As software becomes more pervasive and is used more often to perform critical tasks, it is required to be with higher quality.

The complexities of web services are more complex both in terms of intrinsic and extrinsic complexity values. The better approach to test the web-service is to submit a series of requests to the server and to verify that server responses match the expected return values. The web service interface is a required input to automatically generate a component test for a Web service. Another challenge in testing of web services is that they are built with different user interface levels and tools. Web services can be tested in different levels and phases varies as proof-of-concept testing, functional testing, regression testing, load & stress testing [5] [7]. These are conducted from different perspectives where, the aim of load/stress testing is to find the scalability of the service in terms request fulfillment and transactions, which are dealt in this paper. Load testing is a generic term covering stress testing and a part of performance testing.

Load testing refers to the evaluation of system performance under normal conditions, all the way up to the maximum number of users supported by the given configuration. Load testing provides flexible user scenarios and realistic load types to help application load testing. Load testing helps to guarantee the web applications and web sites that can handle the rate of requests with acceptable performance. Stress Testing is to create an environment with more demands and to observe the performance issues [7]. This work proposes a methodology to define the safety operating point analysis based on Load testing and Stress testing.

The aim of load/stress testing is to find how your web service scales as the number of clients accessing it increases. Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements to determine the load under which it fails and how. Load testing is an evaluation of system performance under normal conditions, all the way up to the maximum number of users supported by the given configuration. Load testing provides flexible user scenarios and realistic load types to help application load testing. Load testing helps to guarantee the web applications and web sites that can handle the rate of requests with acceptable performance [9].

Stress Testing reduces the network traffic and as well as reduce the stress in distributed application. It increases the performance of the software. The idea is to create an environment more demanding of the application than the application would experience under normal workloads. A memory leak is extremely hard to design test cases to detect them. Stress Testing is use to find out memory leaks, bandwidth limits, transactional problems, resource locking, hardware limitations and synchronization problems that occur when an application is loaded beyond the limits.

This paper is organized as follows: Section 2 describes the experimentation methodology carried out in this work. Result analysis and interpretation are done in the section 3 and conclusions are presented in the section 4.

2 EXPERIMENTATION METHODOLOGY

A prototype model of an Online Auction System (**OAS**) is used as a web-application test-bed. This system provides an efficient way to deliver the products to the customers as per the requests received from the customers. The system is implemented with the basic functionalities as follows:

Sample Test program description

1. L= {Input Data (), Validate (), Display ()}, n=n {L} =3

Input Data (): It gets name and password of the users/administrator.

Validate (): It authenticates the users/administrator based on the given input data.

Display (): It displays the users/administrator screen if authenticated, else display the error message.

Where L is defined as Login

2. R= {Input Data (), Validate (), Display ()}, n=n {R} =3

Input Data (): It gets the values such as name, password, address and etc.,

Validate (): It checks whether the given inputs are acceptable. If so, the values are stored in the database.

Display (): It displays alert message if the given inputs are erroneous.

Where R is defined as Registration

3. A= {Input Data (), Store (), Display ()}, n=n {A} =3

Input Data (): It gets auction name, closing date, starting amount for an Auction.

Store (): It accumulates the input values in the database.

Display (): It displays the currently added auction.

Where A is defined as Administrator

4. PC= {Get Input Data (), Display Items (), Closing ()}, n=n {PC} =3

Get Input Data (): The information from the database based on the Closing date

Display Items (): It calls Get Input Data () to gather the inputs and displays the Items

Closing (): It calls Display Items () for update and close the Auction.

Where PC is defined as Process Closing

Testing is a process of executing a program with the aim of finding an error. A good test case is one that has a high chance of finding an as yet undiscovered error. Testing plays a vital role in the life cycle which assures quality of a system. [5] Testing focuses primarily on evaluating product quality, this is realized through the following core practices: Find defects and document defects in software quality, Advise on the perceived software quality, Validate and prove the assumptions made in design and requirement specifications through the demonstration, Validate that the software product works as designed, Validate that the requirements are implemented correctly.

In this experiment set-up, WAPT from SoftLogica LLC [15] is chosen as the testing tool, which has been built with adequate features to test a web application. It is designed such that to reasonably create multiple browsers requesting pages from a web site and to collect statistics on performance regarding the test-bed. It handles dynamic content and HTTPS/SSL, easy to use, support for redirects and all types of proxies, and reports and graphs. It provides the simulation of web server load created by real users. Each virtual user participating in the test is independent from other ones. It can have its own cookies, data and all other parameters. Randomization of delays between page hits and user connection speed emulation let us to create realistic workload against tested server. It also responsible to handle multiple virtual user profiles in a single test scenario, support different language encodings, randomization of delays between page hits, user connection speed emulation, run-time test data generation, support all types of proxy servers, reporting at run-time, provide reports in HTML format, maintain full log on virtual users activities.

3 RESULT ANALYSIS

The primary consideration of the work is to analysis the workload of web based applications for scheduling the request and responses. These analyzes are performed in different user defined scenarios as like ramp-up, fixed and periodic.

Fixed user load: A constant number of users will be made active during the whole test run. All the users will start simultaneously at the beginning of the test.

Ramp-up: A test will be performed where the number of users is gradually increasing; User should select the option with ramp-up number of users and specify initial and final values of interval and some step. It can also possible to set a delay between the virtual users.

Periodic: Periodic user load consists of 2 phases repeating several times during the test; a low load phase and a high load phase. If it is needed to make a periodic load on the tested server, User should specify the number of virtual users for each phase.

The test tool provides the requirements for creating the test scenarios that defines test volumes in terms of number of virtual users participating in the test, test duration, date and time. It also creates the user profile as per the arguments supplied and the profile will be recorded. The Test Execution Parameters are test status, start time, end time, scenario name, Executed person, executed date, duration, no of user, comment. Sessions: Shows the number of user sessions performed during test run for each profile. The Table-1, 2, 3 shows the name of the profile, number of virtual user, number of session, number of pages, no of hits and response time. These parameters are used to increase the performance and also scheduling the request and responses for the system. In the Table-1, 2, 3 show the number of pages responded, hits show the number of pages requested, response time gives the values of response time for each request at various time interval in runtime.

The degree of response can be calculated as

Degree of Response = 100/60 – (60 – Response time per user)

Table: 1 Result of Fixed User Scenario

Profile	User	Session	Hits	Pages	% of Response
OAS	1	4	70	60	98.38
OAS	2	7	126	108	96.19
OAS	3	12	220	190	93.42
OAS	4	13	237	203	90.76
OAS	5	16	292	230	87.60

Figure 1: Graph for Fixed User Scenario

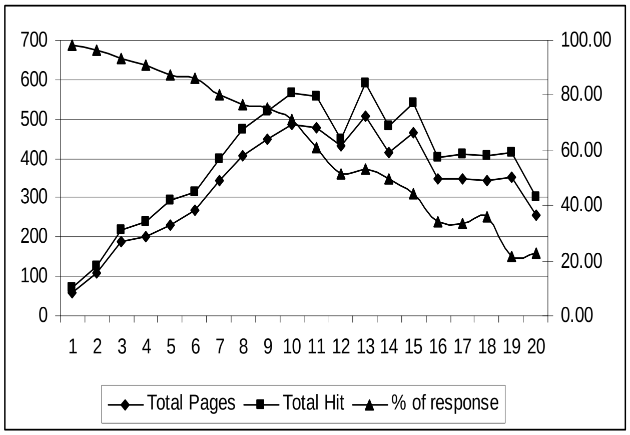


Table 2: Result of Ramp up User Scenario

Profile	User	Session	Hits	Pages	% of Response
OAS	1	3	62	54	98.84
OAS	1,2	5	97	83	93.87
OAS	1,2,3	10	188	162	95.82
OAS	1,2,3,4	7	155	133	96.91
OAS	1,2,3,4,	6	144	122	97.04
	5				

The consolidated information on all user profiles of different test scenario is plotted in the graph as in the figure 1,2 and 3 . These graphs gives the details on number of active users, number of pages processed per unit time and the number of hits supplied per unit time. The number of hits refers to the total number of requests for resources (page code, images, scripts, etc) sent to the server. In the graph, numbers of users are mentioned in the X axis and number of pages and hits are plotted in the Y axis. In the secondary Y axis, degree of response is drawn.

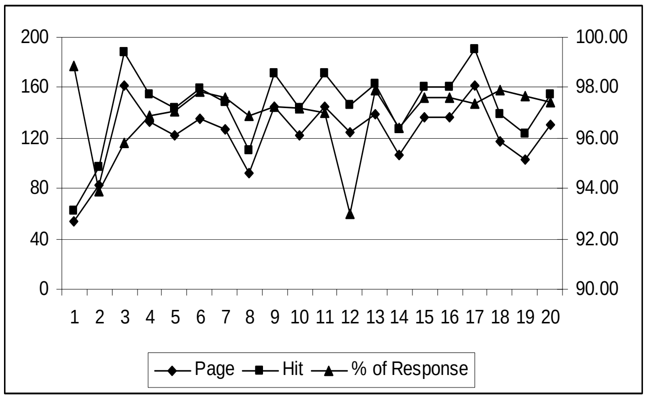
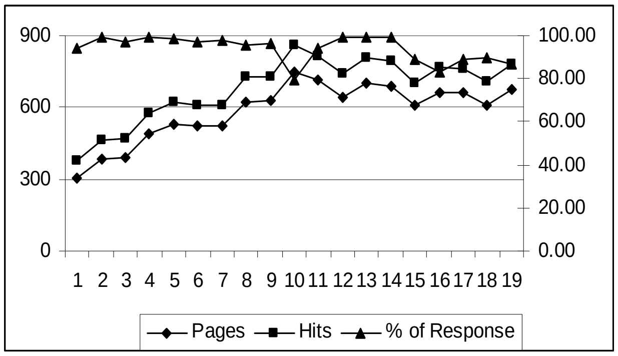


Figure 2: Graph for Ramp up User Scenario

Table 3: Result of Periodic User

Profile	User	Session	Hits	Pages	% of Response
OAS	1,20	9	376	304	94.30
OAS	2,20	14	463	383	98.99
OAS	3,20	16	471	391	96.82
OAS	4,20	24	579	487	99.33
OAS	5,20	27	625	531	98.32

Figure 3: Graph for Periodic User Scenario



Average response time is one of the most important characteristics of load testing; on the other side it measures web user experience. Error tab shows the Timeout error, HTTP error, Socket error in percentage. It will help us to know how the error rate changes during the test when the number of virtual users changes. Error rate is a valuable measure of stress testing to find the maximum number of users that can be served correctly, without errors. The error count includes HTTP Errors, Socket Errors and Timeouts.

The variation in the ability of testing environments is analyzed using the student t-test. The results are analyzed at 5% significance level to achieve the minimum confidence. The experimental set up consists of three sets of independent variables are used.

The intersection points between the response time curve and the other curves as in the graph 1 and 3 gives the insight on the safety operating point region of the test bed web application

4 CONCLUSION

This work has developed for combining the distributed web application and workload to evaluate the performance of the web application based on scenario. An environment is developed for analyzing the performance of the different characteristic user. The WAPT tool used here is suitable for creating the realistic environments for testing the web based services. The tool estimated the growth of the overall traffic, peak loads and duration. Based on the performance issues, the optimal point of operation of the test-bed is constructed. Although this demonstration is in the mere level, it yielded a proof that can attract and encourage the researchers to extend the proposed technique in future.

5 REFERENCES

[1] Bertolino A. and Strigini L. (1996), ‘On the Use of Testability Measures for Expendability Assessment’, IEEE Transactions on Software Engineering, Vol. 20, No.2, pp. 97-108.

[2] Hetzel and William C. (1998), The Complete Guide to Software Testing’, Second Edition, Publication info: Wellesley, Mass.: QED Information Sciences, ISBN: 0894352423.

[3] Joe W. Duran and Simeon C. Ntafos (1984), ‘An Evaluation of Random Testing’, IEEE Transactions on Software Engineering, Vol. SE-10, No. 4, pp. 438-443.

[4] John E.Bentley, Wachovia Bank and Charlotte NC, ‘Software Testing Fundamentals – Concepts, Roles and Terminology’.

[5] Johnson Ryser, Martin Glinz, ‘A Scenario –Based approach to Validating and testing Software System using Statecharts’,12th ICSSEA’99.

[6] Malaiya Y.K., Karunanithi N. and Verma P. (1992), ‘Predictability of Software Reliability Models’, IEEE Transaction on Reliability, Vol. 21, No. 7, pp.539-546.

[7] Mark Lewis – Prazen, Web Service Testing, Web Services Fall, 2006.

[8] Musa J.D., Iannio A., Okumoto K. (1987), ‘Software Reliability- Measurements, Prediction, Applications’, McGraw-Hill.

[9] Osterweil M.S. (1996), ‘Strategic directions in software quality’, ACM Computing Surveys, No. 4, pp. 738-750.

[10] Rothermel G. and Harrold M.J. (1996), ‘Analyzing regression test selection techniques’, IEEE Transactions on Software Engineering, Vol. 22, No. 8.

[11] Rothermel G., Harrold M.J., Ostrin J. and Hong C. (1998), ‘An empirical study of the effects of minimization on the fault-detection capabilities of test suites’, In Proceedings of the International Conference on Software Maintenance, pp. 34-43.

[12] Vahid Garousi,Lionel C.Briand and Yvan Labiche,”Traffic –aware Stress Testing of Distributed Systems Based on UML Models”ACM,ICSE’06.

[13] Victor R. Basali, Richard W. Selby (1985), ‘Comaparing the Effectiveness of Software Testing Strategies’, Technical Report, Department of Computer sciences, University of Marryland, College Park.

[14] Voas J. and Miller K. (1995), ‘Software testability: The new verification’, IEEE Software, pp. 17-28.

[15] WAPT web site : http://loadtestingtool.com