International Advance Journal of Engineering, Science and Management (IAJESM)
ISSN -2393-8048, July- December 2018, Submitted in September 2018, iajesm2014@gmail.com

A Comprehensive Framework to Evaluate Websites

Basavaraj U, Assistant Professor, Department of Computer Science, Government First Grade College and PG Centre Thenkinidiyur Udupi, Karnataka India, Email- ubkottur@gmail.com

Abstract

Websites are essentially client/server applications - with web servers and browser clients. Consideration should be given to the interactions between html pages, TCP/IP communications, internet connections, firewalls, applications that run in web pages (such as applets, JavaScript, plug-in applications) and applications that run on the server side (such as CGI scripts, database interfaces, logging applications, dynamic page generators, asp, etc.). Additionally, there are a wide variety of servers and browsers, various versions of each, but sometimes significant differences between them, variations in connection speeds, rapidly changing technologies, and multiple standards and protocols. The end result is that testing for web sites has become a major ongoing effort [MIC2008]. It is an accepted fact that no system is perfect from view point of performance. Problems pertaining to performance affect all type of systems regardless whether they are client /server architecture or web application systems. The technical factors are the major constraints affecting the performance of the web-based systems. Performance testing of software in general seeks to identify possible bottlenecks and their causes, in addition to optimizing and tuning the platform configuration. It is a testing discipline aimed at verifying an application's ability to operate normally under expected load levels as well as peak load conditions. Determining how well a system scales to enable increased capacity is also an issue. Microsoft Corporation claims that performance testing is about assessing how a system responds to a specified set of conditions and input, and that multiple individual performance test scenarios are required to cover all relevant conditions and input.

Introduction

With the emergence of Web analytics tools early in 2000, data analysis became a strategic element for website optimization. The availability of measures for quantifying website visibility—such as user traffic and behaviors—has resulted in growing strategic importance of these measures for companies. However, having a manager monitor audience behaviors and manage measurement tools is not enough as most websites also use a Web analytic tool. Indeed, it is hard to determine what actions to undertake in order to have a positive impact on traffic analytic indicators. Besides, Web analytics tools are efficient for assessment and optimization of a website. A Web manager's time should therefore be used to respond to new items requiring in-depth, critical thinking to inform strategic decision-making and problem solving. In addition to following Key Performance Indicators (KPI), it is necessary to master the different levers of website performance and, more specifically, the actions that can lead to website optimization. Data collected by Web analytics tools allow advertisers to not only rely on the feeling that this particular lever works properly but also drive business strategy based on concrete data while optimizing operations in real time. This research aims to provide an overview of website performance based on pre-selected criteria as well as show if defined optimization actions have a positive impact on performance. As there are endless ways to define a website's performance, the main objective is to determine a model showing the main topics that illustrate performance. Additionally, there are infinite ways to improve a website. The second objective is therefore to select optimization actions from different sources according to the defined objectives and to test each of these actions and verify if indeed there is a positive impact on said performance.

Websites impose entirely new challenges in the world of software quality. Within minutes of going live, a web application can have many thousands more users than a conventional, non-web application. The immediacy of the web creates immediate expectations of quality and rapid application delivery, but the technical complexities of a website and changes in the browser make testing and quality control much more difficult, and in some ways, more subtle, than "conventional" client/server application testing.

Websites and its functionality is the main show window for potential customers for companies running e-business or e-commerce. Consequences of poor quality in a web context

might be even bigger and therefore, testing plays an important role for good quality web applications. In such systems the quality attributes like robustness, reliability and performance become important properties. A customer surfing the website with a certain goal in mind does not tolerate lack in performance or unacceptable reliability, because there is likely to be a multitude of other companies offering similar services just a mouse-click away. This highlights the importance of making web-based systems to be reliable and perform satisfactory [MIC2008].

To assure website quality, software testing tools and techniques are developed as per the nature of websites and web applications. Automated testing of websites is an opportunity and a challenge. There are a number of automated testing tools available in the market. In this paper an attempt has been made to measure the performance of a website using an automatic web testing software. Two test cases are discussed on various technical grounds to measure the performance.

Performance testing provides with plenty of answers regarding the web based software/ websites to be tested, with the most significant ones are listed below

- Response time
- Throughput
- The maximum amount of concurrent users supported
- Resource utilization with respect to CPU, RAM, network/disk I/O
- Behavior under various workload patterns
- General application weaknesses
- System breaking point – the point where the application stops responding to requests

Response time, also referred to as latency, is the time elapsed until a request has been processed. Response times can be measured on both server and client, where the latter includes the request queue, network latency, as well as the time required by the server to complete request execution. Throughput defines the number of requests that the web system is capable of serving per unit time. Requests per second are the most common measure of unit as far as throughput is concerned. Resource utilization with respect to CPU, RAM, disk I/O and network I/O, represents a cost in system operation. Resource cost can be computed per operation, and is usually measured for a certain user load or distributed on the basis of a workload profile. A workload profile consists of a likely user composition where the users perform various system operations. Simulating simultaneous users can be done by making sure the test scripts incorporate so-called think time. Think time represents the time a user spends between two consecutive requests, for instance when reading web page information or filling out a form. The purpose of think time is to ensure that not all user requests being simulated will occur at the same time. Removing think time from the test script makes sense if the goal is to stress test the web application by simulating concurrent users.

Performance of software is related to its efficiency. According to ISO 9126 efficiency is one of the attributes of software quality. Efficiency has two main aspects viz. time behavior and resource behavior. Time behavior deals with attributes of software that bear on the response and processing times and on throughput rates in performing its function whereas resource behavior deals with attributes of software that bear on the amount of resources used and the duration of such use in performing its function. Thus one finds it logical to map performance to software quality attribute efficiency.

Review of Literature:

Measuring a website's performance is complex because there is a multitude of criteria to define performance. Fortunately, the literature on this issue is plentiful and broad and it is therefore necessary to choose an appropriate model covering relevant criteria which reflect website performance measurement. At first sight, the Awareness, Interest, Desire and Action (AIDA) model seemed to be relevant because it offers a general understanding of the effectiveness of communication endeavors with respect to advertising (Lewis, 1898; Glowa, 2002). This model could be applied to illustrate website performances in the sense that it implies a website must generate awareness, interest, desire, and action (Ber and Jouffroy, 2012). However, given the inherent subjective nature of the desire construct, quantitative

operationalization and measurement through Web analytics is infeasible (Jackson, 2009). Most studies report using the ACT model (Kabani, 2010), a three pillars model based on the Attraction, Conversion and Transformation. However, the ACT model only accounts for the initial attraction of users, but ignores the equally important activity of user retention. Therefore, the REAN model appears to be the most relevant and complete to cover the key research questions (Blanc, Kokko, 2006). Indeed, it takes into account four essential goals—reach, acquire, convert and retain—that define a successful website (Kermorgant, 2008)

CASE STUDY

A website under study has a URL containing number of links that are required on the home page and the size of the page. i.e. http://perftestdomain1/cgi-bin/genAllTypeRandomLinks.pl?links=10andfilesize=10andrandom=0

links => no of links that are required on the page

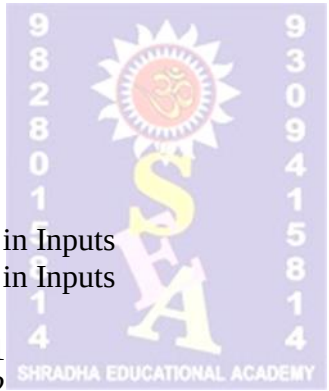
file size => size of the page showing the links

random => It is simply used to avoid URL History rejection of duplicate links placed over Apache and accessible by 25 different domain names logging for the entire site is enabled with bytes sent.

Other details regarding inputs, setup and controlled sets are described with the test cases.

Terms and Notations:

RP : Request Processor
CP : Client process
CU : Crawler/Crawler Unit
dS : RP + CP + CU
FS : File Repository of dS
dSDB : D Server Database
MIDB : MI Database
DB : Data base
M1 : Test Machine 1
M2 : Test Machine 2
I1 : Input Type 1 as described in Inputs
I2 : Input Type 2 as described in Inputs
WS : SQL Server
M1SS : SQL Server on Machine 1
M2SS : SQL Server on Machine 2



TEST I

Inputs:

(a) Local Test Bed: It consists of a combination of hardware and software considered for the testing of website. Our input consists of a URL containing number of links that are required on the home page and the size of the page showing the links. A perl script was created which takes the input in URL.

(b) Controlled Set: The homepage consists of a fix number of hyperlinks say, 25 of the same type say, news having only one further hyperlink. These pages could be divided randomly being pdf, doc, html, xls or ppt types. And other hyperlinks say, 30 having no further links. Entire database server is on one machine. System performance is logged in terms of CPU and Memory and Individual Apps CPU, Memory, Page/Faults Apache Web server Logs for requests, status, url and bytes used.

(c) Setup Details: Two machines were used for this test. Machine 1 called database server having configuration CPU : 2.4 GHz, Memory : 1 GB, OS : Windows 2000 Server + Service Pack 4, Database(DB) : SQL Server 2000 (Enterprise) and Machine 2 : Web Server having configuration CPU : 2.4 GHz, Memory : 1 GB, OS : Windows 2000 Server + Service Pack 4, DB , 20 GBx2 HDD. Apache was running on this machine.

(d) A software which brings the information as per the requirement of the customer is configured by specifying the politeness interval =1 and queue in memory =1501 and logging level is taken to be one to avoid too many page faults.

Outputs:

The various factors relating to performance of website at different timeslots at peak durations and non peak durations obtained after testing are shown as version 1 and version 2 in table 5.1. It is observed that the values performance parameters are on higher side in version 2.

TABLE 1.1: Output of Test I

Performance parameters	Version 1	Version 2
Clock Time in seconds	4741	4801
No of pages downloaded in Clock Time	1964	2969
Total Requests Processed	1964	2966
Total Bytes sent by web server	198858771	300361612
Rate of download = Downloaded Pages/Time Taken	0.414	0.617
Bytes/Second = Total Bytes/Clock Time	41944.48	62562.30
Average CPU Usage (CrawlerUnit)	7.48	2.29
Average Memory Usage (CrawlerUnit)	18.77 MB	16.30 MB
Average Virtual Memory Usage (CrawlerUnit)	98.04 MB	101.90 MB

The following graph shows the comparison of performance of website at peak duration and non-peak durations as version 1 and version 2.

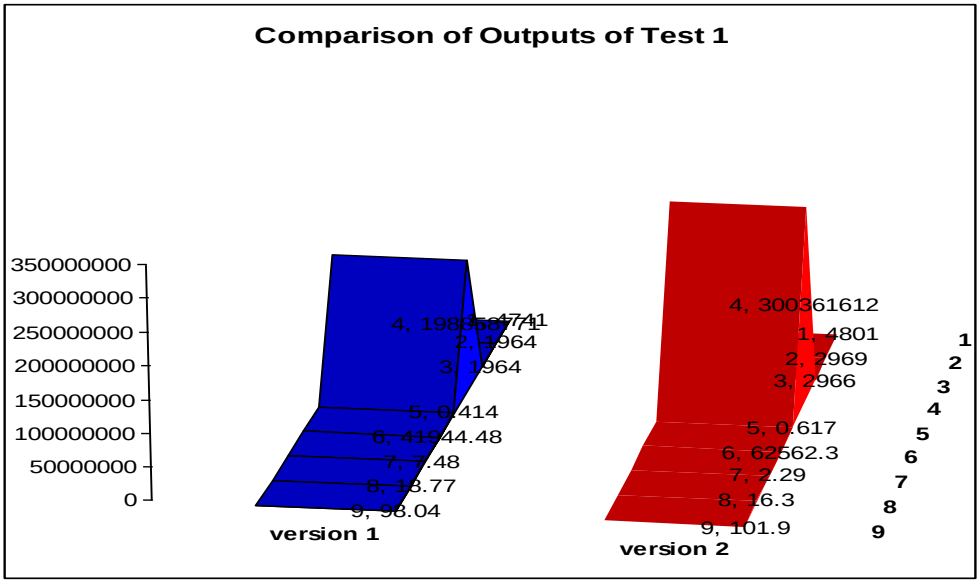


FIGURE 1.1: Comparison of Outputs of test 1

Case 1 Explaination:

Crawler had enough work to do as Master List always had enough entries of different domains. Though the Politeness Interval=10; Average politeness interval per domain, observed in web server log was 34 seconds.

CONCLUSION

Automated web testing ensures that the web applications/web sites/web services' usual functionality works correctly. It provides the ability to reuse and extend the tests across multiple browsers/ platforms/languages/databases/servers and ensure that all the users accessing the web applications get results in an acceptable time. This helps to cut costs, minimize the effort required to test web applications/web sites, increase software quality, reduce time-to-market and use reusable test cases.

Reference:

Aannestad, B., Hooper, J., "The Future of Groupware in the Interactive Workplace", HRMagazine, Vol. 12, Issue 11, November 1997, pp. 37-41.

Abdrabou A, Zhuang W (2006) A position-based QoS routing scheme for UWB mobile ad hoc networks. IEEE J. Select. Areas Commun. 24:850-856.

Agarwal, H., Demillo, R. A. and Spafford, E.H. Debugging with Dynamic Slicing and Backtracking, Software Practice and Experience, 23, pp. 589-616, 1993.
Campos, J., Arcuri, A., Fraser, G. and Abreu, R. Continuous Test Generation: Enhancing Continuous Integration with Automated Test Generation, In the Proceedings of Automated Software Engineering (ASE), 2014.
Camuffo, M., Maiocchi, M. & Morselli, M., 1990. Automatic software test generation. Inform. Softw. Technol., pp. 337-346.
Carnes, P., 1997. Software reliability in weapon systems. , Proceedings of 8th International Symposium On Software Reliability Engineering, p. 114–115.
Canfora, G., Cimitile, A. and De Lucia, A. Conditioned Program Slicing. Information and Software Technology, 40(11), pp. 595–607, 1998.
Cao, Y., Hu, C. and Li, L. An Approach to Generate Software Test Data for a Specific Path Automatically with Genetic Algorithm, In the Proceedings of ICRMS, Chengdu, pp. 888- 892, 2009.
Dufner, D., Kwon, O., Hadidi, R., "WEB-CCAT: A Collaborative Learning Environment For Geographically Distributed Information Technology Students and Working Professionals", Communications of the Association for Information Systems, Vol. 1, Article 12, March 1999.
Edvardsson, J and Kamkar, M. Analysis of the Constraint Solver in UNA Based Test Data Generation, In the Proceedings of the 9th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering, 26(5), pp. 237-245, 2003.
Ehrlich, W. K., Lee, K. & Molisani, R. H., 1990. Applying reliability measurement: A case Study. IEEE Transaction on Software, p. 56–64..