WEBTESTE: A STRESS TEST TOOL

Hugo Saba, Eduardo Manuel de Freitas Jorge, Victor Franco Costa
Nova Tecnologia Informática
Rua Macaubas 520, Ap. 503, Rio Vermelho, Salvador/BA, Brazil

Hernane Borges de Barros Pereira
Fundação Visconde de Cairu - CEPPEV
Rua da Brisa 42, Ap. 1701, Pituba - Salvador/BA - Brasil

Keywords: Stress Test, Web Server, *WebTeste*.

Abstract: The usage of web applications has became a very common activity in the organizations' scope. From the software engineering perspective, the incessant search for production of more robust softwares, with better quality, is a continuous requirement. The main purpose of this paper is to present the *WebTeste*, which is a test tool used to verify the robustness of a web application. After comparison of several simulation's results, the use of distributed and ordered computers suggests more reliable tests. In addition, the analysis of the obtained results can suggest a new (re)design of a web system. The *WebTeste* could be used to perform stress test in order to verify the robustness of a web application more precisely.

1 INTRODUCTION

The success of a test used to identify defects is obtained when improper operations of the analyzed system that is being verified are identified, showing that anomalies there exist (Sommerville, 2000).

In the stress test case, (Bazzana, 2001) argues that location of architectural bottlenecks, impact of hw/sw changes and scalability of the solution are some important aspects with respect to start this kind of test. Because of applications based on web are in continuous evolution, they have peculiar characteristics different from others software categories (Peters and Pedrycz, 2000).

Web applications must attend to a variable number of simultaneous accesses, which will be foreseen by the designer. The stress test is very well applied to the context of some web applications, once this kind of application has requirements such as uninterrupted working (i.e. the application must work 24 hours/day) and attending several numbers of simultaneous requests (Trepper, 2000).

The main goal of this research is to specify and design a test tool, called *WebTeste*, which performs stress test to solve problems related to web application failure. In order to analyze the effectiveness of the *WebTeste*, some simulations based on different scenarios have been carried out.

2 THE *WebTeste* TOOL

The main proposal of the *WebTeste* is to perform stress tests in web applications, generating information that allows software designers to make the best decision for the definition of architectural project. In this way, the obtained information gives support to designers during the official homologation process of the web application architecture.

The *WebTeste* has been designed with the capacity of emulating an unlimited number of users accessing to a web application (i.e. the capacity of realizing simultaneous requests) that will be tested. The *WebTeste* has a distributed and extensible architecture and uses a test plan defined from the system requirements collected during the specification of the web application.

The *WebTeste* also uses as technology to simulate the requests to the web applications, the HTTPUNIT framework, a JUNIT extension (Httpunit, 2004). The *WebTeste* is also composed of an algorithm to simulate the web access in a distributed and ordered way. In this way, *WebTeste* guarantees that the stress only occurs in the computer where the web application is placed, obtaining a more rigorous test. The *WebTeste* allows the designer to be sure about the dimension of the web application architecture.

Considering that the definition of the test plan is one of the major difficulties of test tools, the *WebTeste*

Saba H., Manuel de Freitas Jorge E., Franco Costa V. and Borges de Barros Pereira H. (2006).
WEBTESTE: A STRESS TEST TOOL.
In *Proceedings of WEBIST 2006 - Second International Conference on Web Information Systems and Technologies - Internet Technology / Web Interface and Applications*, pages 246-249
DOI: 10.5220/0001246802460249

is also composed of a module, called *WebMonitor*, that will help in this task. This module, in a interactive and simple way, makes quickly and reliably possible the editing of a test case. Succinctly, this work contemplates (1) the possibility of realizing distributed and synchronized tests, (2) the automatic creation of the test plan by the *WebMonitor* module, (3) the unification of the logs generated by each solicitant computer in the case of distributed tests and (4) a method of the monitoring and analysis of results.

2.1 The *WebTeste* Architecture

During the *WebTeste* development some problems have been identified and some changes have been necessaries.

The *WebTeste* consists of a distributed and ordered architecture composed of four components (Figura 1):

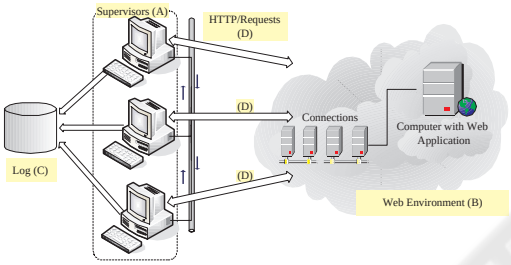

Figure 1: *WebTeste* Architecture.

- **Set of Supervisors** (**A**): Component responsible for the management process of all test. It sends the test plan to the other participants of the test and puts together the logs created by them, generating a centralized log, that is stored in XML format. This allows to guarantee the synchrony among all tests by the supervisors. This component establishes the flow of tests to the web environment;
- **Web environment** (**B**): It can be composed of several elements (e.g. firewalls, web servers in cluster, application servers etc.). The web application, used to the validation of the *WebTeste* architectural model, is placed in this component;
- **Log** (**C**): It represents the centralized log that is generated by the set of supervisors during the simulation of the requests to the web environment;
- **HTTP/Requests** (**D**): This component corresponds to the requests to the web environment.

2.2 *WebTeste* Requirements

WebTeste is composed of functional and no-functional requirements. Functional requirements consist of a set of information (i.e. Creation of Test Plan, Configuration of distributed test and Unification of logs) based on three XML files (i.e. `testplan.xml`, `config.xml`, `logs.xml`).

On the other hand, no-functional requirements are composed of the following five elements: Creation of Test Plan, Distributed Object Remote Method Invocation (RMI), Web Request Framework - HTTPUNIT framework, Algorithm of Synchrony based on Newtop algorithm (Ezhilchelvan et al., 1995) and Test Environment.

3 SIMULATIONS

Several simulations have been carried out. In this paper, the configurations of some of them are presented. These configurations have taken into account three test plans (i.e. three specific tasks run in a web application), two scenarios (i.e. homogeneous and heterogeneous laboratories, in which the distributed tests have been performed), and one centralized test that does not depend on the type of the laboratory, but the computer configurations and the network connections.

- **Test Plans** have been created by the WebMonitor and used in the scenarios #01 and #02 (Figures 2, and 3, respectively). The test plan used in the simulations represents a purchase in a virtual shop and consists of two, ten and twenty requested links towards the purchase conclusion or while the user navigate in web application;
- The main goal of the proposed **types of tests** is to demonstrate the saving of response time when the test is performed in a distributed way with respect to the centralized test, avoiding the stress in the computers that generate the requests:
  - The **centralized test** is performed from a computer that simulates all defined requests to web server;
  - The **distributed test** is performed from two or more computers to web server; each one simulating the half or less of all defined requests. Distributed tests are organized in two categories with respect to the configuration of the computers used in the test: homogeneous and heterogeneous laboratories.

  For this, both centralized and distributed tests have been performed in the two simulation scenarios (Figures 2 and 3) taking into account different quantities of threads in blocks from 100 (minimum quantity) to 2000 with an increment of 100, and from 2500 to 10000 (maximum quantity) with an increment of 500.

3.1 Scenarios

The web architecture for the *WebTeste* simulations has consisted of the Tomcat web server (Tomcat, 2004), version 5.5, working in a computer Intel Pentium Mobile 1.60 gHz with 512mb of RAM and Microsoft Windows XP Professional. In addition, eight computers with different configurations distributed in two scenarios (i.e. homogeneous and heterogeneous laboratories, Figures 2, and 3 respectively) have been used to simulate the supervisors defined in the current version of the *WebTeste* architecture and to perform the stress test in a web application.

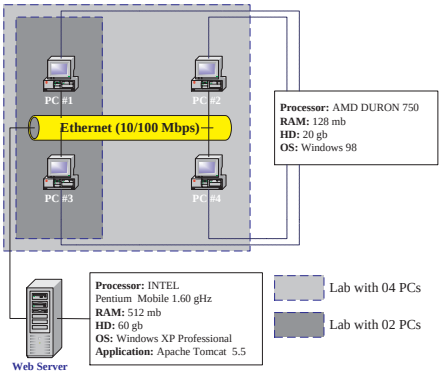

Figure 2: Simulation scenario #01 for the *WebTeste* performing: Homogeneous Laboratory.

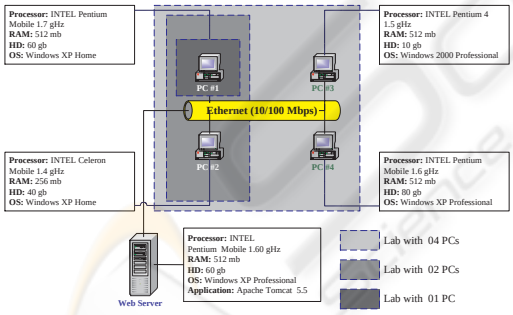

Figure 3: Simulation scenario #02 for the *WebTeste* performing: Heterogeneous Laboratory.

The web application used to perform the tests consists of a computer system that virtually simulates a computer store. In this system, none kind of database persistence is used, only in memory.

3.2 Results

After the simulations, the results of the centralized and distributed tests have been obtained. It is important to (re)comment, that in the centralized test only one computer has been used to perform the supervisor's function. On the other hand, in the distributed test, two and four computers have been used as supervisors.

Figure 4 depicts a comparative chart of the averages response times (ms) obtained from the *WebTeste* simulations taking into account the scenarios #01 and #02 for the test plan composed of two links. The arrow in the Figure 4 indicates that has occurred some problem during the performance of the last set of threads (i.e. 10000) in the homogeneous laboratory with 02 computers. This problem can be related to the traffic flow of the used LAN and/or to the low computer configuration (i.e. AMD Duron 750 processor, RAM 128mb).

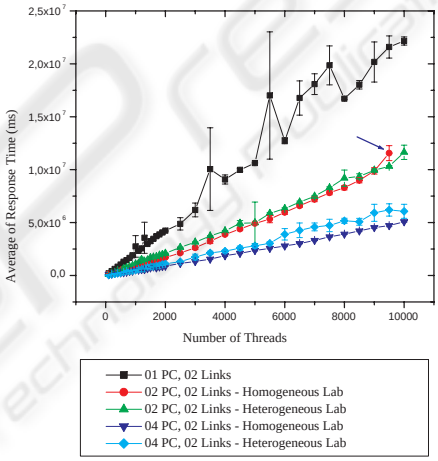

Figure 4: Comparative Chart with respect to Averages of Response Times for the *WebTeste* simulation scenario #01 and #02 for the test plan composed of two links.

As can be observed, the distributed test has been more efficient than the centralized test, due to while the threads number started in the centralized test for each client (i.e. supervisor computer) increased, the response time also considerably increased. When the test has been performed in a distribute way, it has been noticed that the same request numbers had a lower response time, because of the fact of each supervisor computer starts a lower thread number.

Moreover, another interesting analysis can be carried out fixing the number of computers. For instance, Figures 5 and 6 respectively show the comparative charts of the average of response time (ms) obtained from the *WebTeste* simulations considering the centralized tests and the distributed tests composed of four computers. Two problems are indicated by an arrow in the Figures 5 and 6:

- The first one refers to the arrow in the Figure 5

that indicates the occurrence of some problem during the performance of the last set of threads (i.e. 10000) in the centralized test for 10 links. This problem is only related to the traffic flow of the used LAN, once the computer configuration used (i.e. Intel Pentium Mobile 1.7gHz, RAM 512mb) is upper than the computer configurations used in the homogeneous laboratory (i.e. AMD Duron 750 processor, RAM 128mb);

- The second one refers to the arrow in the Figure 6 that indicates the occurrence of some problem during the performance of the last three set of threads (i.e. 9000, 9500 and 10000) in the distributed test for 04 computers. This problem is not only related to the traffic flow of the used LAN, but also to the low configuration of a computer used in the heterogeneous laboratory (i.e. Intel Celeron Mobile 1.4gHz, RAM 256mb). This situation indicates that a low computer configuration in a heterogeneous scenario can reduce the performance of the stress test.

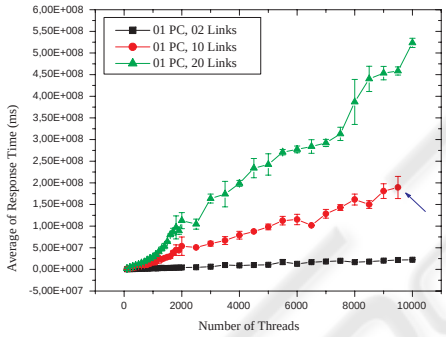

Figure 5: Comparative Chart with respect to Averages of Response Times for the *WebTeste* simulation based on the test plan composed of one computer and two, ten and twenty links.

From the obtained results, it has been evident that the thread numbers started in the tester computer can have an influence on the response time of the web environment.

4 CONCLUDING REMARKS

The current version of *WebTeste* performs ordered and distributed tests and takes into account the fault-tolerance processing, the facility of the test plan design, and the monitoring and analysis of results.

The obtained results from the *WebTeste* tool application have been interesting. *WebTeste* has presented efficiency during the distributed and synchronized tests, due to the significant reduction of the average of response times (Figures 4, 5 and 6).

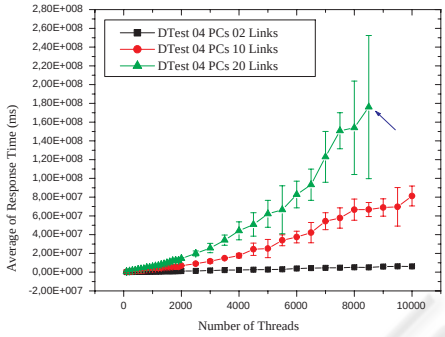

Figure 6: Comparative Chart with respect to Averages of Response Times for the *WebTeste* simulation scenario #02 for the test plan composed of four computers and two, ten and twenty links.

The stress test using the *WebTeste* throughout the design and implementation of web applications will can contribute with the robustness and reliability of these applications, improving their quality.

As future research activities, it is proposed to perform the stress test based on *WebTeste* tool taking into account access to the database in order to verify some aspects such as the relation between the stress test tool and the database persistence, and/or the variation of response times.

REFERENCES

Bazzana, G. (2001). Ensuring the quality of web sites and e-commerce applications. In Wieczorek, Martin and Meyerhoff, Dirk, editor, *Software Quality: State of the Art in Management, Testing, and Tools*, chapter 13, pages 178–191. Springer-Verlag, Berlin.

Ezhilchelvan, P. D., Macêdo, R. A., and Shrivastava, S. K. (1995). Newtop: A fault-tolerant group communication protocol. In *International Conference on Distributed Computing Systems*, pages 296–306.

Httpunit (2004). Http unit. http://httpunit.sourceforge.net//.

Peters, J. F. and Pedrycz, W. (2000). *Software Engineering: An Engineering Approach*. John Wiley & Sons, New York.

Sommerville, I. (2000). *Software Engineering*. Addison-Wesley, London.

Tomcat (2004). Apache jakarta tomcat. http://jakarta.apache.org/tomcat/.

Trepper, C. H. (2000). *E-Commerce Strategies*. Microsoft Press, Redmond.