Efforts Required For Performance Engineering of a Web Application

Dr. K.K. Aggarwal, Dr. Yogesh Singh & Ms. Vandana Gupta

ABSTRACT

NewPort group research states that 52% of the Web applications fail to scale-they don't perform acceptably under real-world usage. Performance testing is normally considered as a last stage in the development lifecycle. In some cases it is taken up as a reaction to customer complaints. This approach normally results in significant design changes in the final stages or a trade-off with the functionality or robustness of the application. Considering the factors like shorter delivery cycles, high costs of rework, competition etc., it has become necessary that the performance testing activities are properly planned to ensure a strict adherence to the customer requirements. Testing should hence be considered as an activity that needs to run in parallel to the design and development activities. All these need proper planning to be done and necessary resources allocated for performance engineering right at the beginning of the SDLC.

This paper describes the need for performance testing, what all is done as part of the exercise, what all resources are required and ball-park % of total SDLC effort required for performance testing. Currently there is no methodology available for estimating the effort required for Performance testing of web applications. The ball-park estimate for performance testing in this paper can be used as a starting point for the same.

INTRODUCTION

Conventional application software testing is well defined with a set of methods and established over the years (William Perry, 1995). Each method is implemented with a proper strategy. However, one can use some heuristics to test the specific part, which cannot be done by the available methods due to inherent practical aspects of the software. But, the conventional methods may not work with the Web based applications. These applications vary with functionality, presentation, and target users. Thus, Web applications are dynamic in nature and require strong testing methodology. Web testing mainly involves testing functionality of the system like in conventional system (Pressman & Roger S), presentation layer and performance (Carolyn Duffy Marsan). Performance testing occupies the dominant role in testing life cycle of the E-commerce applications. During testing of the Web applications, many issues must be addressed as the web site is subjected to many unknowns and uncertainties (Daniel A. Menasce and Virgilio A.F. Almeida. 1998).

One of the main objectives of performance testing is to maintain a Web application with low latency, high throughput, and low utilization. Provisioning the right infrastructure for the software systems is critical, as bad performance leads to user dissatisfaction, productivity impediments and lost business.

The performance testing (Menasce D.A. and Dowdy L.W., 1994) is a must in case where performance is the key "Critical To Process" parameter for the application. In a situation where client is very particular about the performance and the feasibility of the web application in a low bandwidth dial up environment and wants the application with low latency, high throughput, and low server utilization. There are situations when on a severely constrained bandwidth of 128kbps client need a response time of less than 8 seconds for pages with rich User interfaces.

Where application performance is a must to have non functional requirement, right from day one a proper due diligence needs to be done in architecting the solution, by building very stringently reviewed and tested application blocks and by conscious implementation of best practices right from the design phase.

Currently the biggest problem is that performance testing is not given the due importance during the initial planning phase of the web application development (Edward Lazowska et al., 1984). The required resources in term of software, hardware and human are not budgeted for the same. Owing to these shortcomings mostly web applications fail to scale in production environment and customers generally complain of poor performance. The paper presents a comprehensive list of activities to be performed as part of performance testing and percentage of total SDLC effort required for performing the same.

PERFORMANCE TESTING PROCESS

Performance improvement for the project was carried out in three phases (Dr. Subraya BM and Subrahmanya SV). These are

* Discovery Phase
* Modeling Phase
* Execution and Post Execution Phase

DISCOVERY PHASE

* **Workload Modeling**

Workload modeling is performed to capture user behavior patterns, business productivity targets and workload growth. The Workload model identifies the transactions associated with the application, user loads and the concurrent occurrences of these transactions. It captures the variation of these user loads in normal and peak scenarios, different service levels and the workload growth over a period of time. The network, the protocols, the hardware configuration of the existing systems and current levels of utilization are also part of this model. This model describes all factors that have a bearing on performance and capacity. Identifying these parameters was very important as this model forms the basis for all subsequent infrastructure decisions.

The model also captures the workload distribution on the system during a typical business day. A Workload Distribution is a representation of the functions performed by a user community on a system. The target performance goals are worked collaboratively between performance team and all key stakeholders through mutual experience, conversations and workshops. Testing and tuning continues until all goals were met.

MODELING PHASE

* **Test Strategy and Planning**

Based on the Workload modeling document a testing strategy is developed and all the resources (hardware, software and human) are identified and timelines are arrived at.

* **Hardware Setup**

This requires defining the testing architecture with machine and network configuration in terms of the Number and configuration of each type of server database, application, and web. The hardware setup should be very similar to proposed production setup and it would be best if the same were performed on the production environment.

* **Software Setup**

Load generator systems are required to generate the workload for the required number of users. Appropriate test tool needs to be selected for the performance testing. All the test tools come with stringent virtual user license terms and cost of the tool is directly proportional to the number of virtual users required.

* **Test Design and Development**

This is the stage in which the actual load generation scenarios and scripts will be generated. The details of script generation will be arrived at on completion of the previous stage and on identification of the load

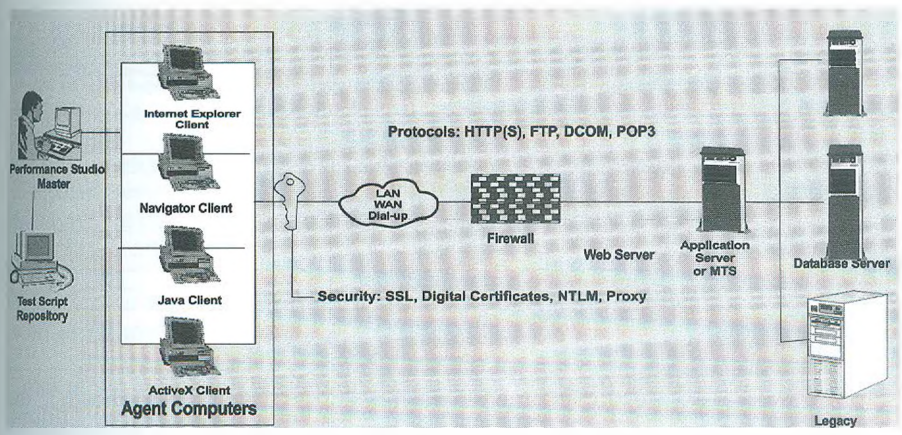

Fig. 1 A typical Performance Test Environment setup for Web Applications

generation and monitoring methodology. In this phase, different scenarios are identified using the workload model. Workload indicates the transaction mix; the scenarios are designed for the peak workload pattern and for the isolation scenario where each transaction is run independently. The scripts that depict the scenarios are recorded and parameterized. Think times captured in the transaction model are incorporated in the script.

In order to test the scenarios; the data required should be set up e.g. creating user ids for all the users, creating configuration data for each user. This is needed to avoid processing and accessing the same data, which results in unnecessary caching or locking at the database. Also the data needed for the performance test should be significant to get realistic results.

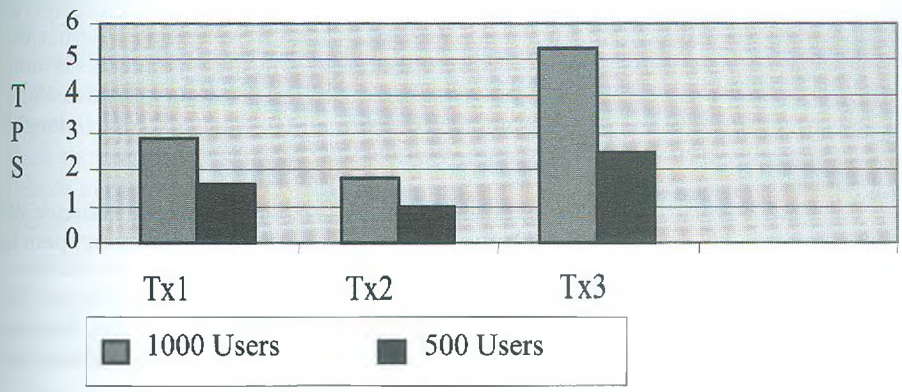

Fig.2 Transactions Per Second

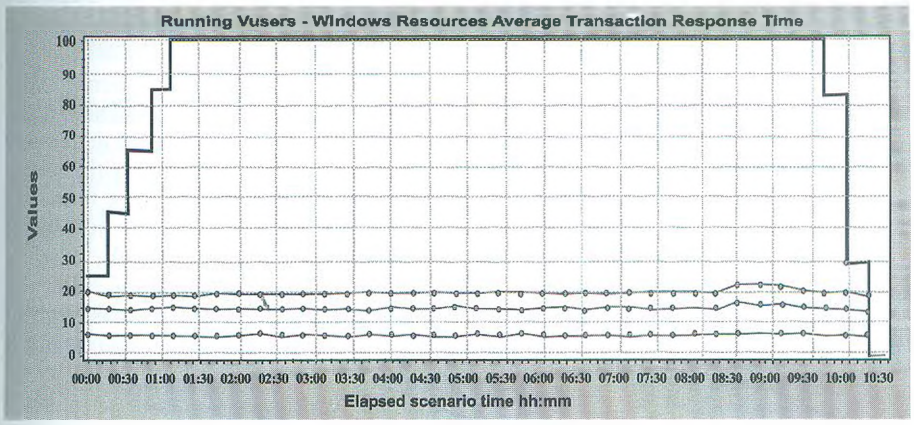

Fig. 3 Response Time Vs Number of Users

EXECUTION AND POST EXECUTION PHASE

Execution occurred in an iterative fashion. During the initial stages of testing, tests and analysis continues until required bottlenecks are identified. Once required tuning is done, then the application is re-benchmarked and tests are repeated incrementally until either the target goals are met, or further required tuning is identified. In some cases additional tests are executed to assist developers/architects in correctly identifying and tuning bottlenecks. This entire process is repeated until no further tuning was possible and both the <stakeholders> and <Performance Engineering Team> agree that further testing and analysis is not required at this time.

Following tasks are performed

* Test execution and results generation
* Test results analysis and tuning recommendations
* Post tuning test reruns
* Report generation

MEASURE PUT A WHAT & WHY

Some of the key windows performance counters to be measured are Requests Queued, Requests per second, Requests totally executed, Request Execution Time, %Processor time, %time spent by application process and database server processes. Some of the Inference we can make from the numbers are:

Scalability This can be inferred in 2 different ways (James A. Whittaker).

One will be to check whether the number of server pages per second increase close to linearly as the number of users was increased as shown in the Figure 2. This is used for capacity planning (Ralph Barker).
Second is to check whether the

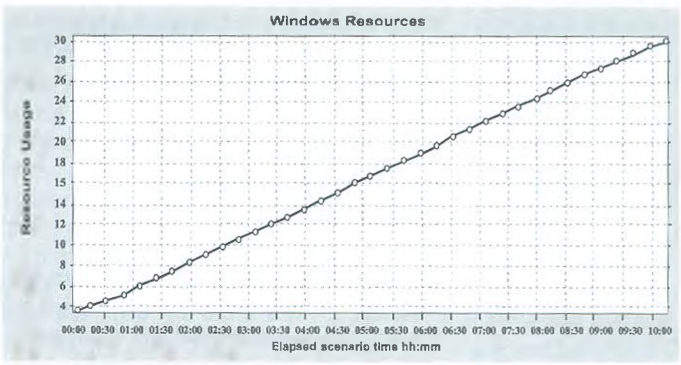

Fig. 4 Load on Different Servers

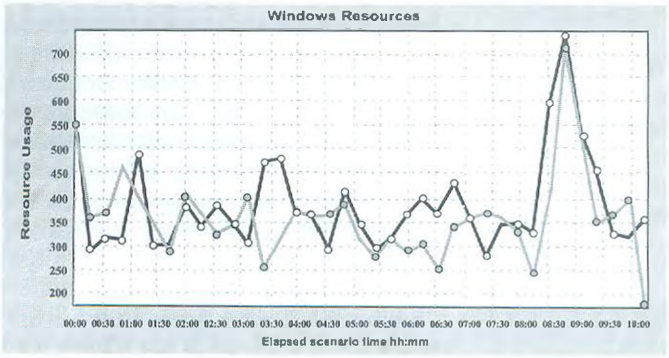

Fig. 5 Time Spent at Servers Vs Network

Fig. 6 Transaction Response Time

	W1	W2	W3	W4	W5	W6	W7	W8
	13-Jun	20-Jun	27-Jun	4-Jul	11-Jul	18-Jul	25-Jul	28-Jul
Create Performance Testing Scripts								
Data Base Set Up								
Performance Testing Round 1								
Fine Tune Environment								
Performance Testing Round 2								
Review and Sign Off Perfromance Testing Results								

Fig.7 Project Plan for Performance Testing

Table - 1

Task	Description	Deliverable	Effort (Person Da
Workload modeling	Prepare workload description of system	Workload model	20
Test environment setup	Setup load generation servers, application, measurement tools ...		10
Scenario Creation	Define scenarios for load testing	Test workload plan	25
Test script preparation	Record the test scenarios in scripts	Test scripts	25
Test data generation	Prepare test data for use in load testing		10
Test execution	Run the tests (steady state) and measure	Measurements	50
Bottleneck analysis	Identify bottlenecks		20
Rerun Tests	Rerun the tests		50
Test report preparation	Prepare test report containing measurements, deployment architecture and bottlenecks	Test report	10

response time is invariant across user loads as shown in t figure 3. This is used in case the customer is focused response time.

In figure 3, the % of users is the parameter which is steppi up and then coming down at the end of the test. The straig lines in the diagrams are the response times for varic transactions. As the line is straight, it indicates that t response time does not vary as % of users increases.

Web farm load balancing We need to be sure that t requests are getting load balanced across the web servers we had planned. We can do this by checking whether t Requests Total counter across both the web servers is m or less the same. The figure 4 below shows a web farm wit machines which has more or less been evenly load balanc by Network Load Balancing Service (NLBS).

Time spent at the server versus time spent in the network need to know on an average how much time is being spen the server versus the network. Based on this, we will kn whether we have to focus on tuning the server piece or t bandwidth piece. This can be known by the Requ Execution time counter of Windows. The figure 5 shows t value in a Web farm scenario with 2 web servers.

The graph in figure 5 indicates that the time spent in t server is varying from about .3 seconds to about .8 secon across the time span of the test. Please note that the test t shows only an average of value for each time interval. I quite possible that there are requests, which take 2 seconds in the server but are not shown in the graph due the law of averages.

Response time variation analysis As mentioned above, t average request execution time does not reflect accurat as to whether all the transactions are finished within tin To find out how many exception scenarios were there, could check the response time variation i.e. the number hits for each response time as shown in the figure 6.

The graph in figure 6 shows that the response time is more less around 7 seconds for most hits and for very few hits has gone past 10 seconds.

PROJECT PLAN FOR PERFORMANCE TESTING

The project plan in figure 7 depicts the different tasks of a performance testing phase and effort and elapsed time spent in each phase.

The discovery phase of performance testing starts early in the SDLC where during the requirements and design phase workload modeling is done.

Table 1 gives the total effort required for performance testing of a web based application.

Total effort spent = 220 Person Days = 220/21 = 10.47 Person Months

These are the actual figures for schedule and effort for the performance testing exercise carried out for a 280-person month effort .Net based web application.

As it is evident from figures above the total performance engineering effort comes out to be approximately 4% of the total project effort.

Generally this effort is not budgeted when initial project planning is done but as web applications are becoming complex day by day, performance testing is must for the success of any web application and needs to be performed before application is rolled out to the world.

CONCLUSION

Among many complexities, performance testing is one of the difficult activities in E-commerce application development life cycle, which has to be tackled with more vigor and aggression. As Web applications are becoming more and more complex, testing of Web applications occupies the dominant role in the software engineering life cycle particularly performance Testing. Performance of many Web sites depends on the load on the site at peak time under varying conditions. Performance testing is normally conducted in a reasonably simulated environment with the help of performance testing tools. This paper describes the need for performance testing, what all is done as part of the exercise, what all resources are required and ball-park % of total SDLC effort required for performance testing. Currently there is no methodology available for estimating the effort required for Performance testing of web applications. As it is evident from figures above the total performance engineering effort comes out to be approximately 4% of the total project effort. The ball-park estimate for performance testing in this paper can be used as a starting point for the same. A proper planning needs to be done for the performance testing phase of the project and all resources need to be budgeted and planned for the same.

REFERENCES

1. Carolyn Duffy Marsan, Study cites *"Net backbone for slowing Web sites"*, Network World, www.nwfusion.com/archive/2001/119957_05-07-2001.html
2. Daniel A. Menasce and Virgilio A.F. Almeida. (1998), *"Capacity Planning for Web Performance Metrics, Models, & Methods"*, Prentice Hall, PTR, New Jersey
3. Edward Lazowska et al. (1984), *Quantitative System Performance*, Prentice Hall Inc., New York
4. James A. Whittaker, *"What is Software Testing? And why is it so hard?"*, IEEE Software, Jan/Feb 2000
5. Menasce D.A. and Dowdy L.W. (1994) *"Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems"*, Prentice Hall, NJ.
6. Pressman & Roger S. *"Software Engineering: A Practitioner's Approach"*, McGraw Hill International.
7. Ralph Barker, *"Optimizing Internet Bandwidth"*, Performance Computing, January 2000.
8. Dr. Subraya BM and Subrahmanya SV, *"Performance Testing: A Methodical Approach to E-commerce Application"*, 13th International Quality Week 2000 Software Quality Week 2000 Internet Quality Week 2000, May 30 - June 2, 2000, Hyatt Regency, Embarcadero, San Francisco, California.
9. Dr. Subraya BM and Subrahmanya SV, *"Object driven Performance Testing in Web Applications"*, Proceedings of The First Asia-Pacific Conference on Quality Software, Hong Kong, October 30-31, 2000.
10. William Perry. (1995), *"Effective Methods for Software Testing"*, John Wiley, New York