ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st&22ndMarch, Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Performance Testing With Realistic Load Testing Scenarios & Load Test Data

C.Prathiba1,S. Jayanthi, Dr.P.S.K.Patra N.Ezhil Arasu

Department of Computer Science , Agni College of Technology, Chennai , Tamil Nadu, India.

Assistant Professor, Department of Computer Science , Agni College of Technology, Chennai , Tamil Nadu, India.

Head Of the Department, Department of Computer Science , Agni College of Technology, Chennai ,.Tamil Nadu, India.

Performance / Security Consultant, South East Toyota, Capgemini, Chennai, Tamil Nadu, India.

Abstract—for a good valid Performance testing cycle, user load simulation and monitoring the health of the application is the major part. User load simulation is determined by number of users, list of transactions & test data. Instead of conducting traditional way of selecting the Test Data & Transaction path, this paper explains best way to select the test data & Transaction Path closely to match with Production Environment. This paper compares a traditional load test results with the new load test which we followed our new procedures to choose Test data & Test Path and finally concludes that our approach is more effective for an existing Application Performance Testing.

Keywords—Software Performance, Workload Profile, Load Model, Performance Test Data, Performance Test Scenarios, Performance work Estimation & Modeling.

I.INTRODUCTION

Many of the web applications undergo rapid development lifecycles and have very little time frame to complete all activities during Plan and Develop.

However, many web based applications will fall in performance since it was not properly tested against correct load while it was in Testing Environment. There were many reasons behind and most important factors are:

1. Poor Test Planning
2. Not covering all Critical Transactions
3. Incorrect Load Numbers for Performance Testing

Not Covering all paths Random Test Data No End to End Monitoring of all partsPoor Performance Testing will lead to failure of application in production / live environment [1] which will lead to loss of revenue, loss of customers & their trust on the application.

Most of the applications, performance is measured in base of Transaction response times, Error Rate and Transaction Failure Rates. During requirement gathering phase, Performance testing expectations for each page / Transaction will be gathered and the user numbers expected to access

each pages / transactions will be fixed as Non Functional requirements.

To avoid these failures due to the mistake done by the Performance Engineers we need to adopt few procedures at each level of performance testing from Requirements gathering to Performance Reporting. Here are the few areas where we can improve our practice:

1. Identifying Test Scenarios (Test path)
2. Identifying Test Data
3. Calculating Load Profile

Section II will determine the process of redefining the transaction path, Section III will determine the process of redefining the test data, section IV will give the results of a load test with these methods applied, in, section V will give the comparison between this results with previous test results, Section VI will give concludes this exercise.

I.

II. Transaction Path Design for an Existing Application

Consider an existing application is being replaced with new application or consider an existing application enhancement project. Here we have a concrete idea about present users and load on the production servers. We took an application which was used by dealers for an automobile Giant in United States. This application supports all dealer related transactions with the automobile company. It supports not only United States, it also extended to Canada, South America & Arab Countries.

We can start this process by collecting all valid submit, search & Update operation in the application. This will give us the overall transactions to monitor during a performance test. Almost we collected the data for all accessed transactions and pages for a year period to analyze the transactional behavior. We took the help of the Business Team of the Application team to come up with the list of transactions. We made them into different work groups based on the Type of the users & Frequency of usage.

After getting list of transactions from application, now we started gathering data from production servers. We got the information metrics from these following ways.Log analysis (Web server Access Log Files)[5][8] & Google Analytic kind of tools (Omniture this case)[7]

We got a bunch of Junk data from these files. The big task is now to analyze. We finalized our threshold based on our average number of page loads happened in a peak day. How a peak day identified will be in the coming paragraphs. We fixed a threshold of 3% of Average Load in a peak day for dealer transactions and 1% of Average Load in a Peak day for Corporate user based Transactions. For example to eligible for selected list of transactions for performance Testing, a transaction should exceed more than 300 hits in an hour for a dealer transaction and 100 in case of a corporate user based transaction. Usually one year data will give a good idea about this. We took an average of # of times accessed for 6 month of data excluding Saturday's, Sunday's and national holidays since including these days in our average calculation might bring down the cut off or threshold value.

At next steps, we find the Peak Day for the whole website. This comes from an Omniture Metrics which listed the access counts for Entire site for the selected period. From that graph, we identified the maximum hits that have been accessed throughout the year. This Peak Day hits is used in our Threshold calculation. Next our Task to find the Individual Transaction's Peak Days. This is required, since it is not necessary to be true that an individual Transactions Peak Day and the whole Websites Peak Day to be same. We found this truth while browsing through the Raw Data that different Transaction had their Peak numbers in different Days in a year.

After gaining a complete final list of Load test transactions, now our recommendation will start. In this paper we are recommending to get the transaction names *& transaction path from production test.

Next goal is to identify the peak transactions per hour data.

Following are the options available to pick from.

1. Peak number of transactions within a peak hour window on the peak day for the specific transaction
2. 90th Percentile of transaction data over a given interval
3. Average of maximum 'N' number of transactions over a given interval (1 month).
4. Maximum number of transactions over a given interval.

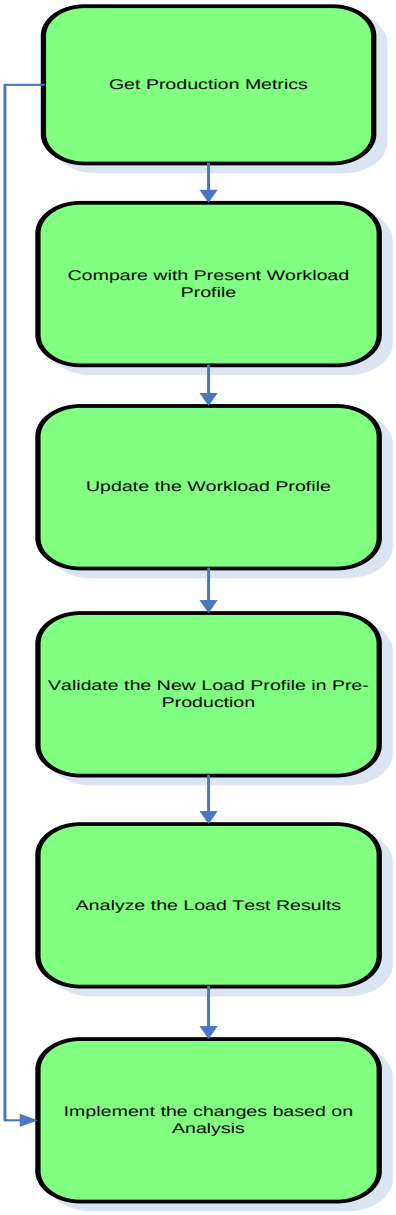

Fig. 1. Process flow for Work load Profile remodeling.

Approach 1: Peak number of transactions within peak hour window on the peak day for the specific transaction is an aggressive approach which always yields higher number. There could be many reasons for a transaction to reach a peak in an hour of 1 peak day. We are considering only one hour data from one year which may not be accurate and may be skewed. This will be always more than the peak load and less than or equal to double peak load

Approach 2: 90th Percentile of transaction data over a given interval will account for load of all peak hours in a year. This will be close to the peak value. If this number is too large compared to the average value, we can go with 80th percentile.

Approach 3: Average of maximum 'N' number of transactions over a given interval is taking the average of hourly hits over one year. This average should exclude non peak days like holidays, Sundays and night time and system down times. Otherwise, the calculated number will be very less load on a given day.

Based on the suggestions and advantages, we took the approach 2 to find the 90th Percentile value of transactions to find the numbers.

III. Test Data Match with Production

Performance test data is the data set used for executing the transactions using virtual users during a load test [11]. By carefully analyzing the usage data from production application, we can redefine and parameterize the test data to 95% accuracy. Setting up performance test data is the most critical task during performance test runs since it controls the throughput, and the performance of the back end systems. It also has an impact on the application server in case of xml to xslt conversion or any kind of sorting done on the server side[12].

Below are the factors to decide a better Test data[3]:

1. Size
2. Complexity
3. Test Cases need to be run
4. Variety in Test Data
5. Coverage of Business Rules
6. User Type from all access Levels
7. Data from all regions
8. Data for Multiple browser Emulations.

Test data is the important during Performance Test runs since it controls the throughput, and the performance of the back end systems. For small systems, validation of all data is possible to find the right data for performance testing[13]. But in case of Huge volume of data it is impossible to analyze all the data to find the right test data. That activity will be a time / cost consuming activity. Sampling[6] is one approach which can be adopted when the data is voluminous. Note that sampling does not mean that we were not equally interested in all the items in the population.

For example, earlier performance tests done for a search transaction to retrieve less than 10 orders. But the production test data clarifies that 70% of users will retrieve at least 25 orders per search. This test data need to be taken care

In addition to that for a good performance test, data volume plays a significant role. Even we have a good test data, but if we don't have correct data volume, then it will produce wrong response time results. If the data volume is low, we may not be able to find many hidden bottlenecks in the query and database side.

For example, we served for a major automotive client in US. During that we have a functionality called VIN Look up for last 8 digit VIN numbers. Due to environment constraints we were not able to load much data into the Vehicle information database. During Performance test cycle, we have good results. But when the system open for end users, on the day of deployment itself all VIN lookups started failing which forced us to back out the deployment in 8 hours.

This example leads to two main flaws. One is there is not enough data volume in the simulated environment. The second is 8 digits VIN number search is not prioritized in our load test.

An important aspect of performance testing is that the set of test data used must be very close to ***'real' or 'live'*** data which is used on production[4]. The following question arises: 'Ok, it's good to test with real data, but how do I obtain this data?' The answer is pretty straightforward: from the people who know the best – the customers. They may be able to provide some data they already have or, if they don't have an existing set of data, they really helped us by giving feedback regarding how the real-world data might look like. In case we are in a maintenance testing project we could copy data from the production environment into the testing bed. It is a good practice to anonymize (scramble) sensitive customer data like Social Security Number, Credit Card Numbers, and Bank Details etc while the copy is made.

II. Results of a Load Test with these procedures

We finally ran into our Performance Testing lifecycle with the said application (Dealer Portal for an automobile Giant in United States).

We took the list of transactions from Step 2 and good test data from Step 3. We take the number of transactions required for identified transaction names from step 2 to do our performance tests.

Please find the below sample data sheet which has identified transaction & its target number of Transactions in Table 1.

Here is the summary of Load Profile[10] Changes we did:

Based on the transaction data captured from Omniture,

1. Load on Locate Vehicle Module reduced
2. Load on DV Modules increased
3. Load on OV Modules slightly decreased, but the distribution of load between OV modules are adjusted
4. Load on MI Modules doubled and we have 40 users with 5 sec think time and earlier, we used 32 users with 10 sec think time
5. For R&T and P&F, the load is not changed, but added a few more reports included in the DE II baseline test

Key Business Scenario / Transactions	Target Non-Peak Response Time (secs)	Target Peak Response Time (secs)	2013 Q4 Peak Txn/hr (Revised)	2012 Q4 Peak Txn/hr (Previous)
OV02_CONFIGURE_LANDING_PAGE	0.258	0.34	2110	1117
OV02_ALLOCATION_SELECTED	1.351	1.62	2110	1117
OV02_GET_DATA	4.029	4.99	2110	1117
OV03_LOAD_OPTIONS_PAGE	3.373	4.42	2110	1101
OV03_CHOOSE_OPTION	3.472	4.16	1987	2379
OV04_CONFIG_SUMMARY	0.431	0.60	2152	1096
OV03_VALIDATE_CONFIG	1.582	1.87	2152	1096
OV04_SUBMIT_PRELIM_ORDER	0.991	5.30	1449	1078

Table 1. Transactions per Hour Expected

Here are the changes we did in the test data:

1. All data used is for 2013 MY
2. OV allocation groups are spread between small, midsize and SUV vehicles
3. In OV, all Sell sources used in PT test data file
4. All region (US/CA/MX/IPC) data used in PT scripts

Here are few points we observed from these tests:

1. Some Unusual Transactions went high during the test
2. We Effectively found a STRUCK Thread issue in WEBLOGIC Server
3. We found the issue with JDBC connection with Mainframe DB2 Tables.
4. We found INGRES OUTBOUND Connections errors during these tests.
5. These three issues identified as Critical performance Defects which were not identified in earlier tests which are a major achievement for our new approach of Performance Testing.

IV. Comparison of Performance Test Results with Previous Cycle

We compared the test results with previous test results for the same application. The difference is huge since we were able to find existing three critical defects along with a major change in the load & response pattern.

In our both load tests, there are no environment hardware / software changes and in same sized network. The tool used to produce the load also same in this case.

Here we can take a look at the HTTP Summary of both Load Tests:

Here is the HTTP Summary of Present Load Test:

HTTP Responses	Total	Per second
HTTP_200	3,576,010	662.715
HTTP_302	64,640	11.979
HTTP_304	2,418	0.448
HTTP_404	19,400	3.595
HTTP_500	1	0.000

Table 2. HTTP Return Code Summary Latest Test

And here is the HTTP Return code summary of Previous cycle Performance Test for your eyes.

HTTP Responses	Total	Per second
HTTP_200	2,573,930	452.281
HTTP_302	37,325	6.559
HTTP_304	1,297	0.228
HTTP_404	10,649	1.871
HTTP_500	28	0.005

Table 3. HTTP Return Code Summary of Previous Test

The huge difference in HTTP Return Code 200 is telling the story of previous load test didn't load the application much expected. Since the HTTP return code 200 per second in Production environment is ranging from 400 to 600 per second.

And from other part, there is response time degradation at an average of 1 sec observed in all transactions which could be due to the new pattern which will unlock the next phase of analyzing the root cause. The new set of runs unveiled a transactions list for the architecture team to work on them. Here is the comparison of the response time of these transactions:

Transaction	Previous Cycle		Present Load Test	
	Avg Response	90 % Percentile Time	Avg Response	90% Percentile Time
OWB_MI02_VIEWINV_PRELIM	1.089	1.326	15.625	42.119
OWB_29_RT_METRICSREPORT	NA	NA	11.728	18.834
OWB_MI02_VIEWI_INTRANSIT	1.089	1.326	7.117	15.439
OWB_PF04_REGISTRATIONS	1.089	1.326	5.189	15.243
OWB_MI02_VIEWINV_ALL	1.089	1.326	6.909	15.24

Table 4. Response Time Comparison Table

V. Conclusion

By comparing the results, we can conclude that the new approach of identifying the Transactions, and its path, Test Data and load profile numbers has numerous advantages over the traditional Load Testing approach. It was able to discover the new defects in the application and able to match close with the real world scenarios. From the point of Time and cost, the new approach will take more time and more skilled works and yields better results. For a small & Straight forward applications it will not find enough time and budget, but for big & complex applications these approach need to be followed.

As a result of these, the client asked us finish the testing with their one more critical application and later they will make these mandatory for their bunch of 256 applications.

By Overall, this exercise aims to educate the business owners & the Project Development teams about the present load pattern and gives enough room to correct the work load profile before failures.

VI. Literature Survey

Modeling a Realistic Workload for Performance Testing
Christ of Lutteroth, Gerald Weber:

Form-based applications range from simple Web shops to complex enterprise resource planning systems. Draheim and Weber adapt well-established basic modeling techniques in a novel way to achieve a modeling framework optimized for this broad application domain. They introduce new modeling artifacts, such as page diagrams and form storyboards, and separate dialogue patterns to allow for reuse. In their implementation they have developed new constructs such as typed server pages, and tools for forward and reverse engineering of presentation layers. The methodology is explained using an online bookshop as a running example, in which the user can experience the modeling concepts in action. The combination of theoretical achievements and hands-on practical advice and tools makes this book a reference work for both researchers in the areas of software architectures and submit-response style user interfaces, and professionals designing and developing such applications.

References

[1] Modeling A Realistic Workload For Performance Testing Christ Of Lutteroth, Gerald Weber

[2] Mercury Interactive Corporation. Load Testing To Predict Web Performance. Technical Report Wp-1079-0604, Mercury Interactive Corporation, 2004

[3] Daniel A. Menasc´E. Load Testing Of Web Sites. Ieee Internet Computing, 6(4):70–74, July 2002.

[4] Performance Testing For Web Based Application Architectures (.Net Vs. Java Ee) Osama Hamed1 And Nedal Kafri2

[5] Omniture Available At *Www.**Omniture**.Com/En/Products*

[6] D.A. Menasc´E, "Load Testing, Benchmarking, And Application Performance Management For The Web," In Proc. 2002 Computer Management Group Conference, Pp.,271-281, Reno, Nevada, December 2002. Available At:Http://Www.Cmg.Org

[7] Google Analytics Available

[8] Weblog Analyzer Tool Available To Download From *Indihiang.Codeplex.Com/*

[9]Ucml

[10] Creating Effective Load Models For Performance Testing With Incomplete Empirical Data, Scott Barber, Performance Test/Analysis Consultant

[11] Scaling For E-Business
Daniel **A.** Menascc

[12] Meier, J.D.; Vasireddy, Srinath; Babbar, Ashish; Mackman, Alex (2004) "Improving .Net Application Performance And Scalability", Http://Msdn.Microsoft.Com/Library/Default.Asp?Url=/Library/En-Us/Dnpag/Html/Scalenet.Asp, Last Accessed June 7,2004

[13] Analyzing Load Test Results And Errors In The Tables View Of The Load Test Analyzer - Http://Msdn.Microsoft.Com/En-Us/Library/Ms404656.Aspx