International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 12 (2020), pp. 4746-4750

Soak Testing of Web Applications Based on Automatic Test Cases

Rijwan Khan[1], Ayman Qahmash[2] and Mohammad Rashid Hussain[3]

[1]Professor, Department of Computer Science and Engineering, ABES Institute of Technology, Ghaziabad, Affiliated to AKTU, Lucknow, India. ORCID: 0000-0003-3354-3047.
[2]Assistnat Professor, Department of Information Systems, King Khalid University, Abha, Saudi Arabia.
[3]Assistnat Professor, Department of Information Systems, King Khalid University, Abha, Saudi Arabia

Abstract

Software testing is a procedure of ensuring to deliver fault free software. Different types of testing are applied in the software industries to insure the reliability of the software. In this paper our main focus is on performance testing. The performance testing is also a kind of software testing in which the different areas have been covered like load testing, smoke testing, soak testing and stress testing etc. In this article soak testing has been performed on different parameters. The core performance is considered when businesses are at its highest (peak) by its hits.

Keywords: Software Testing, Performance Testing, Test Cases, Load Test, Soak Testing, Smoke Testing

I. INTRODUCTION

Nowadays, all the people want a very fast web application but at the same time for each web application they also concern about its reliability. On clicking any web application, it will go to its web page. When the business is on the peak, then there are so many facts about the web application. At the peak hours of the business so many people access the web application so it also concerns the system performance. How much load can be put on a web application? How many people can be permitted to access the web application at same time? The heap execution gives out the reaction seasons of all the significant business basic exchanges in a web application. In the event that the information base, application worker, and so forth are additionally checked, and afterward this basic test would itself be able to point towards any bottlenecks in the application programming load testing is normally directed in a test climate indistinguishable from the creation climate before the product framework is allowed to go live. The performance testing is concerned with load, smoke, stress and drench testing [1, 2, 10, 11].

II. PERFORMANCE TESTING TYPES

A different kind of performance testing is proposed in research [5, 6, 7, 8, 9]. When someone checks the performance testing then these given below types of the testing are discussed.

II.I Smoke or Sanity Tests

After the test load profile is made, a 'Smoke or Sanity tests' on the contents will be executed to guarantee if the contents and the application setup are done accurately. This is a must for any sort of execution test. Any deformities recognized in the dry-run will be fixed during this stage. Smoke test or sanity tests are run with 1 client for each content unexpectedly. In the case of everything looks great, at that point a genuine remaining task at hand is joined to see whether the application is steady under pre-characterized load.

II.II Load Test

In load testing the number of the users are suggested by the client or it can also decide by the performance team depending on the number of transactions to be achieved. While the test is running then for the load testing the following thing should be monitored in different levels e.g. tool level, application server level, database server level. Load testing is observed through controllers and these observations are noted for every test conducted [1, 2, 3].

II.III Stress Tests

Stress testing is one sort of testing. Stress testing is additionally a type of performance testing which used to decide the maximum furthest reaches of limit inside the application. [1, 2, 4].

II.IV Soak Tests

Soak Testing, otherwise called perseverance testing, is generally done to decide whether the framework can support the ceaseless anticipated burden. The douse testing screen memory usage. The throughput and reaction season of the framework is resolved from start to finish of the application run.

II.V Fail Over Tests

After the significant load, stress and soak tests are performed, failover tests are led which decides when the application smashed does and this occurs in a joint effort with the creation of uphold groups and information base teams [10, 11].

II.VI Scalability Tests

Scalability testing is a kind of testing where it has been observed that application is steady under the guaranteed load, when clients are thusly expanded.

III. METHODOLOGY

The approach received for the performance testing can be shifted generally yet the goal for performance test continues as before. All the most significant level performance testing is quite often led to address at least one danger identified with cost, opportunity costs, progression and additionally corporate

standing [12, 13, 14, 15, 16]. The center performance testing exercises are given in figure 1.

Fig 1. Process for Performance Testing

IV. TEST TOOL AND UTILITY

The test tools and their performance are given in table 1

Table 1. Tools and Utilities

Tools	Performance
Load Runner 11.0	The tool is utilized to •Captures end-client business measures and makes a robotized execution testing content, otherwise called a virtual client content. •Organize, drive, oversee, and screen the heap test. •Create the heap by running virtual clients (VUsers) • View, analyze, and think about the exhibition results.
HP ALM	To schedule and run the test for customer's prerequisites for different clients.
Perceiver	To analyse the performance bottlenecks, if any. This would be used to monitor CPU and memory utilizations, other counters that would be monitored would be Disk utilization, process queues, JVM out of memory exceptions etc.
Quality Center 10.0	To raise performance defects and to manage them.
Perfmon Logs	To analyse the hits on the server for each request
CA Wiley	To break down the exhibition bottlenecks, assuming any. This would be utilized to screen CPU and memory usages, different counters that would be observed would be Disk use, measure lines, JVM out of memory special cases and so on.

V. EXPERIMENTAL SETUP

V.I Soak Run

The target of this soak test is to execute 24 hours throughout the day test for the application under test with application groups running all the cluster loads, measure occupations that run during the creation run of the application and decide any presentation bottlenecks like high CPU usage, memory use or any equipment issues. Likewise, disappointments would be additionally thought about and would guarantee that application is steady all through the 24 hours term. In this period, spikes accordingly times and CPU usage in workers would be passed on to application groups. The test would run with the same client heap of 160 clients.

Table 2. Soak Test Transaction Response Time

Transaction Name	Average Transaction Response Time	90% Response Time	Passed	Failed
Case Creation Submit	3.234	3.851	2183	5
View Note Final	2.312	2.987	2778	17
Edit Note Final	3.998	4.211	3195	1
Increase Counter Submit	2.331	2.734	5211	4
Reduce Counter Submit	2.988	3.129	5216	0
Close Case Final Submit	4.453	4.811	2010	2
Display Page View	3.4	3.654	3113	4
View Alerts Info	4.112	4.42	365	1
File View Image	0.965	1.341	92132	156
Scan Report	0.413	0.521	1	0
Manual Report	0.312	0.432	1	0
Print Report	0.212	0.255	1	0
Alerts Report	1.256	1.656	1	0
Page Report	1.525	1.721	1	0
Cases Report	1.012	1.318	1	0

V.II Observations

Average exchange reaction times and 90 percentile reaction occasions were under the SLA (administration level arrangement) of 5 seconds.

- All the focused-on volumes for every business usefulness were accomplished.
- Load Balancing to every worker (2) were similarly conveyed (affirmed by the worker groups).
- All the disappointments were because of Data Issues however since the disappointments were under 10%, so it was under worthy cutoff points.

- Central processor Utilization on the application worker, which was prior an issue with the smoke test, got settled and use was ideal for the heap test.

Table 3. Soak Test Transaction Load Balancing

Server	Number of Requests
Application Server 1	130232
Application Server 2	134515

Table 4. Soak Test CPU Utilization

Server	Avg. CUP Utilization	Max CPU Utilization
Application Server 1	50.08	69.05
Application Server 2	48.98	88.61
Database Server	70.12	82.23

Table 5. Soak Test Memory Utilization

Server	Avg. CUP Utilization	Max CPU Utilization
Application Server 1	51.38	89.05
Application Server 2	54.68	76.88
Database Server	45.15	70.34

V.III Performance Statistic

The table vi features the exhibition screen logs arranged on the application workers to recreate the heap test results features to coordinate the tally accomplished during the testing.

Table 5. Load Test Performance Monitor Table

Label	Hits	Avg	Min	Max	Active	Avg Active
/RMC/ClosedTicket.jsp	1888	924	180	1908	18	13
/RMC/Filenet/Image/src	99912	51230	1890	99989	1390	100
/RMC/Case/Create.jsp	6123	3443	189	6442	27	12
/RMC/Increment.jsp	2898	2101	24	3212	23	3
/RMC/Redue.jsp	2991	1890	198	3001	45	9
/RMC/Blank.jsp	231314	178654	30432	234456	9870	230
/RMC/Index.jsp	30098	21321	980	31234	344	24
/RMC/goCount.jsp	13123	10231	1340	12897	30	30
/RMC/ChangeStatus.go.jsp	2345	1234	32	2567	18	10
/RMC/Controller.ms.jsp	1488	1010	10	1879	8	2
/RMC/Details.go	1648	1001	18	1890	10	1

V.IV Load Runner Analysis of Graphs

In figure 2 soak test with virtual users are mentioned with the graph. Every user run the system successfully when even peak hours. The application performed all needed activities during the soak testing. Run all business functions that it is intended to do. Now it is Soak Test Vusers ramp up minutes.

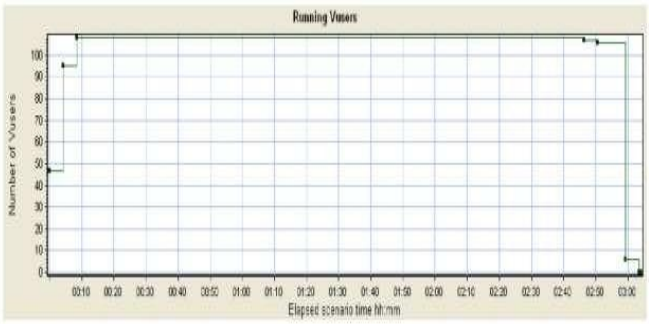

Fig 2. Soak Test Vusers ramp up

V.V Throughput

In figure 3 the server returns a number of bytes during soak testing. In figure 3 a graph shows the stability of the test during soak test period when no spikes are placed. Few spikes were seen during the soak test which was though not sharp, so it was acceptable.

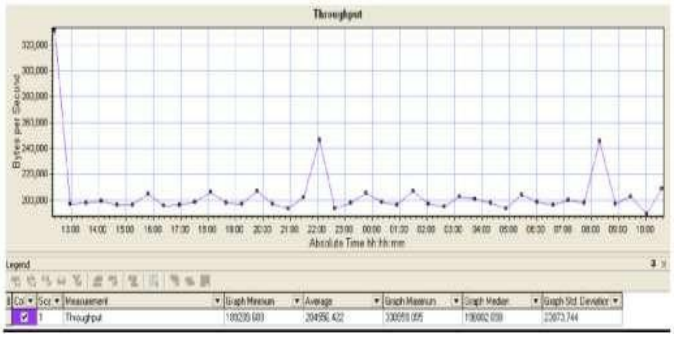

Fig 3. Soak Test Throughput Pattern

V.VI Hits Per Second Graph

The graph given in figure 4 is for performance testing. Soak testing hits per second pattern is shown in figure 4. Here consistency with the throughput is measured. Once all the users are completely ramped up then the pattern becomes stable. On an average, 17 hits per second were observed.

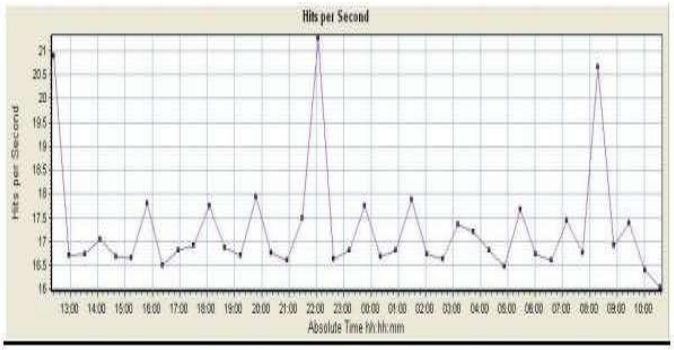

Fig 4. Soak Test Hits Per Second Pattern

V.VII Server Inbound Message Rate

Please find below the server-inbound message rate which shows at the peak the server addressed more than 1100 messages at a time for Filenet requests as it is targeted for more volumes adjusted for soak test (less volume targeted) and more users were taken to test this functionality.

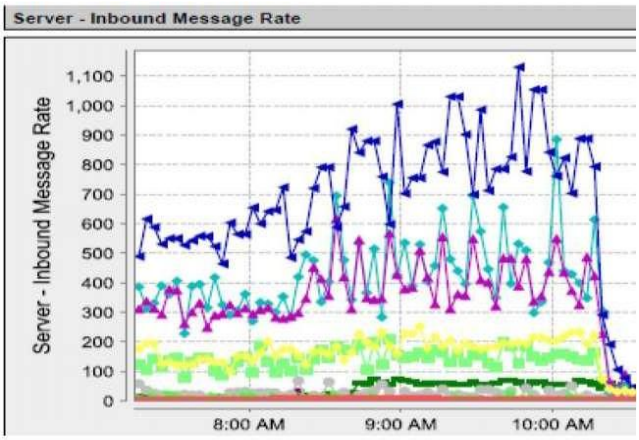

Fig 5. Soak Test Server Inbound Message Rate

V. VIII Server Outbound Message Rate

Please find below the server-outbound message rate which shows at the peak the server responded more than 1100 messages at a time which is in sync with what we have observed in our inbound messages graphs. So, no pending messages were seen during the test so no hung threads were observed.

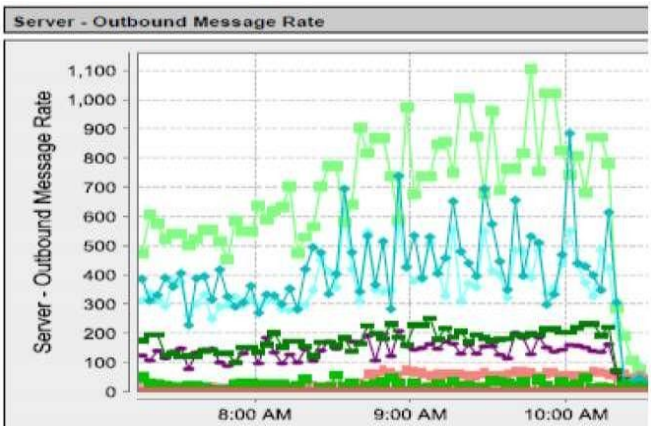

Fig 6. Soak Test Server Outbound Message Rate

V. IX Web sphere Report-Hung Threads

As it can be seen 0 hung threads were observed during the test.

So, the test ran fine without any issues.

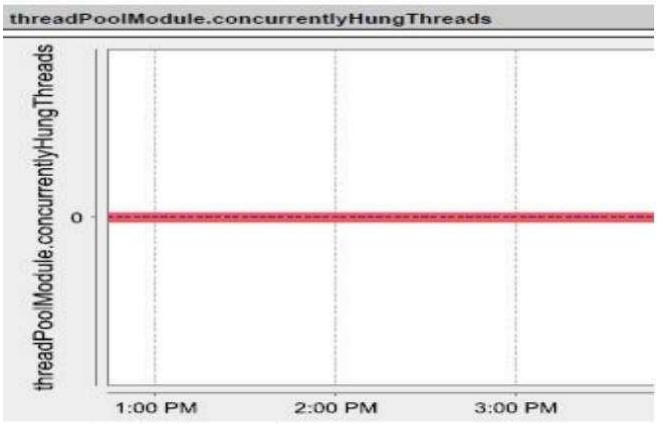

Fig 7. Soak Test Server Thread Pool Concurrent Hung Threads

VI. CONCLUSIONS

In this paper all processes for load testing have been done without any issue. The load test went fine during the soak testing or load testing. The average response times or pass/fail percentages of load tests went fine. More than 10% failed transactions count as the failed test but we did not observe such issues in our load test either, so we got the go ahead from the application teams to go for the SOAK test. Though we certainly observed more CPU and memory utilization but it was accepted and approved by maintenance, application, and server teams since they executed many other things parallel to the soak test like running batch loads, maintenance and clean-up jobs which occupied high CPU and memory.

ACKNOWLEDGEMENT

We would like to express our gratitude to Deanship of Scientific Research, King Khalid University, Kingdom of Saudi Arabia for funding this work.

REFERENCES

[1]. Zhu, Kunhua, Junhui Fu, and Yancui Li. "Research the performance testing and performance improvement strategy in web application." 2010 2nd International Conference on Education Technology and Computer. Vol. 2. 2010.

[2]. Hislop, Helen, Dale Avers, and Marybeth Brown. Daniels and Worthingham's muscle testing: Techniques of manual examination and performance testing. Elsevier Health Sciences, 2013.

[3]. Beizer, Boris. Black-box testing: techniques for functional testing of software and systems. John Wiley & Sons, Inc., 1995.

[4]. Beizer, Boris. Software testing techniques. Dreamtech Press, 2003.

[5]. Kit, Edward, and Susannah Finzi. Software testing in the real world: improving the process. ACM Press/Addison-Wesley Publishing Co., 1995.

[6]. Hetzel, William C., and Bill Hetzel. The complete guide to software testing. John Wiley & Sons, Inc., 1991.

[7]. Kuhn, D. Richard, Dolores R. Wallace, and Albert M. Gallo Jr. "Software fault interactions and implications for software testing." Software Engineering, IEEE Transactions on 30.6 (2004): 418-421.

[8]. Pearl, Judea. "Heuristics: intelligent search strategies for computer problem solving." (1984).

[9]. Verbauwhede, Ingrid, Patrick Schaumont, and Henry Kuo. "Design and performance testing of a 2.29-GB/s Rijndael processor." Solid-State Circuits, IEEE Journal of 38.3 (2003): 569-572.

[10]. Weyuker, Elaine J., and Filippos I. Vokolos. "Experience with performance testing of software systems: issues, an approach, and case study." IEEE transactions on software engineering 12 (2000): 1147-1156.

[11]. Ward, C. L., et al. "Design and performance testing of quantitative real time PCR assays for influenza A and B viral load measurement." Journal of Clinical Virology 29.3 (2004): 179-188.

[12]. Menascé, Daniel. "Load testing of web sites." Internet Computing, IEEE 6.4 (2002): 70-74.

[13]. Zhao, Nai Yan, and Mi Wan Shum. "Technical Solution to Automate Smoke Test Using Rational Functional Tester and Virtualization Technology."Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International. Vol. 2. IEEE, 2006.

[14]. DeMillo, Richard A., et al. "An extended overview of the Mothra software testing environment." Software Testing, Verification, and Analysis, 1988., Proceedings of the Second Workshop on. IEEE, 1988.

[15]. Mahajan, Manish, Sumit Kumar, and Rabins Porwal. "Applying genetic algorithm to increase the efficiency of a data flow-based test data generation approach." ACM SIGSOFT Software Engineering Notes 37.5 (2012): 1-5.

[16]. Khan, Rijwan, and Mohd Amjad. "Automatic generation of test cases for data flow test paths using K-means clustering and generic algorithm." International Journal of Applied Engineering Research 11.1 (2016): 473-478.