A STUDY ON PERFORMANCE EVALUATION AND OPTIMIZATION OF E-COMMERCE SYSTEMS

[*]D. Saravana Prakash[1], T. Santha[2]

[1]Research Scholar, Department of Computer Science, Dr. G.R.Damodaran College of Science, Coimbatore

[2]The Principal, Department of Computer Science, Dr. G.R.Damodaran College of Science, Coimbatore

Abstract

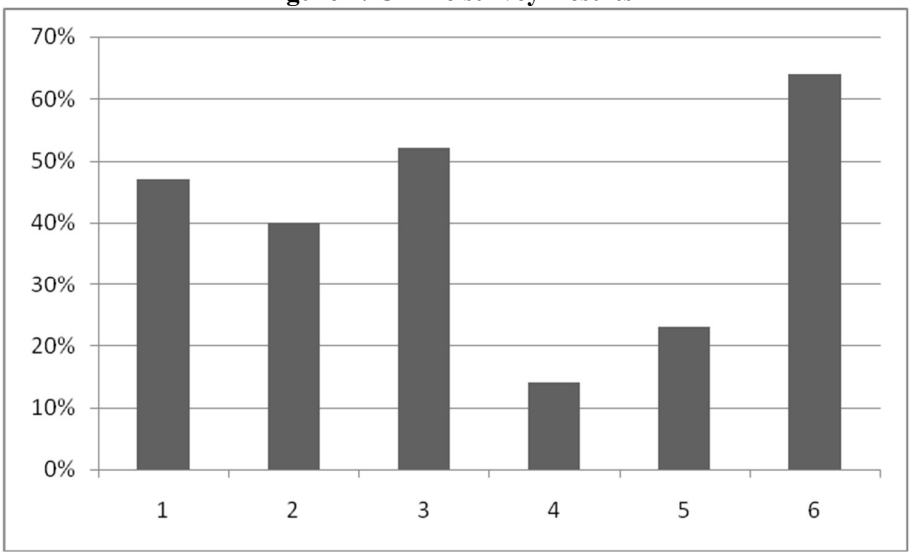
The performance of an e-commerce system plays a vital role in converting a visitor into customer and optimizing an e-commerce system is an extraordinarily complex task as it involves several factors that are interrelated. Though, there exist several approaches and strategies to optimize the performance of e-commerce systems with specific focus on a particular area or factor, achieving an overall performance optimization of the system with conservation of all factors is remaining a complex and challenging task. This paper studies and compares numerous factors that affect the performance of an e-commerce system, various approaches, and techniques existing for evaluating and optimizing the performance of e-commerce system.

Keywords: **Performance Evaluation, Performance Optimization, Performance Managements, E-Commerce Applications, Factors Influencing Performance**

1. INTRODUCTION

The goal of any e-commerce system is to convert a visitor into customer and this low conversion rate, in general, is due to lower performance of the system. The results of an online survey (as shown in Figure 1) conducted by media group recently shows that:

Figure 1: Online survey Results



1. 47% of people expect a web page to load in two seconds or less
2. 40% will abandon a web page if it takes more than three seconds to load
3. 52% of online shoppers claim that quick page loads are important for their loyalty to a site
4. 14% will start shopping at a different site if page loads are slow
5. 23% will stop shopping or even walk away from their computer
6. 64% of shoppers who are dissatisfied with their site visit will go somewhere else to shop next time

These numbers clearly show that the success of an e-commerce system totally depends on its performance. But, optimizing the performance of an e-commerce system is one of the most challenging scientific processes as it involves many complex and closely related factors. Once the optimum performance is achieved, it has numerous advantages and a few of them are as follows:

- Improved performance allows several people to browse stores and make purchases at once without slowdown.
- Dynamic website acceleration allows for the fast delivery of constantly changing content such as stock levels and new product information.
- Shopping cart acceleration ensures that customers aren't left waiting at the final hurdle. With many users abandoning shopping carts if loads times are too slow, this can help boost sales and limit lost opportunities.
- Streaming technologies can be used to provide rich advertising and promotional media to website visitors.
- Stores, warehouses, and head offices can be connected via high-speed networks to ensure that all stock levels are kept up to date and new product information can be disseminated quickly.
- Analytics information can be used to inform future product decisions.

The traditional approach to developing mission critical systems that are scalable (i.e., systems that can handle significantly heavier workloads than the ones for which they were initially designed) involves extensive performance testing of the target architecture with realistic workloads. Simulation programs have been successfully used to reduce testing costs in several industries. Some specific examples are semiconductor [1], networking [2], telecommunications [3], aerospace, manufacturing, and pharmaceutical [4]. The very fundamental methodology of performance testing named Hypothesis Testing and Ranking was actually derived from the above four simulation models [5].

In this paper, we performed a detailed study on performance testing to help configure the application for high scalability. We have found numerous performance evaluation and optimization methodologies that are in use to enhance the effectiveness of performance testing. One of the key objectives of performance testing is to uncover problems that are revealed when the system is run under specific workloads. This is inherently a dynamic activity that involves running the software system on selected inputs. We usually learn from performance models by extrapolating from the modeled behavior, while the most useful information from performance testing are unexpected and unacceptable behaviors. Since this behavior is not supposed to be present in the architecture, it is unlikely to have been included in the model.

There are many number of models, methodologies, techniques, suggestions and proposals to evaluate the performance of a general web applications [18, 19]. But, a typical e-commerce sites are complex, consisting of hundreds of machines with a large number of software configuration parameters that may take many different values. The number of possible combinations of the values of these parameters can be extremely large. The performance of e-commerce sites significantly depends on the proper setting of these parameters. In addition, interaction effects between parameters and the types of requests made to the system also impact the site's performance.

Therefore, it is imperative that one be able to determine the sensitivity of an ecommerce site's performance to the various configuration parameters. Given that it is not feasible to test all possible combinations of configuration parameters, the past researches have provided practical and ad-hoc experimental methodology based on statistical techniques to 1) identify key configuration parameters that have a strong impact on performance, 2) rank the parameters in order of their relevance to performance, according to criteria design, and 3) provide an algorithm to perform interaction analysis between the various parameters and the types of requests submitted to the site. This paper studies the factors that have significant impact on the performance of the system and various models in use to evaluate and optimize the overall performance of the system.

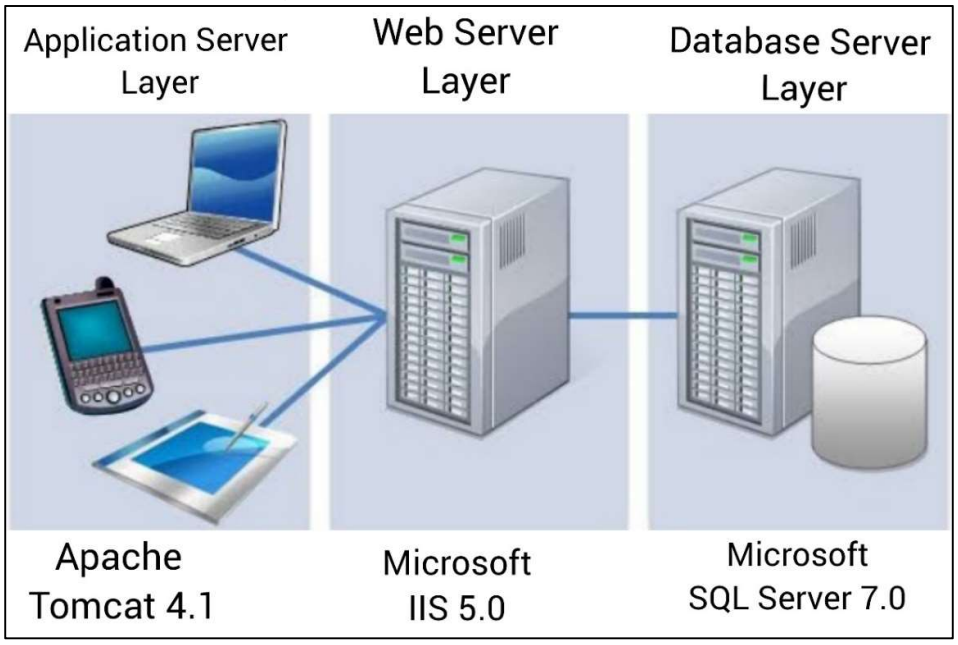
2. DESCRIPTION OF FACTORS THAT AFFECT PERFORMANCE

Twenty-eight factors from three-tiered e-commerce site architecture are widely chosen and considered for the performance based experiments. The following list briefly describes the factors, organized by the layer in which they reside and sorted in an alphabetical order. As shown in Figure 2, the web server layer uses Microsoft Internet Information Server (IIS) 5.0, the application server layer uses Apache Tomcat 4.1, and the database layer uses Microsoft SQL Server 7.0. Factors 1-13 are the Web server factors, 14-16 the application server factors, and 17-28 the database server factors.

1. Application Optimization: Whether to allow performance optimization of only the foreground applications (more processor resources are given to the foreground program than to the background program), or all applications (all programs receive equal amounts of processor resources) [5].

2. Application Protection Level: Whether applications are run in the same process as Web services (low), in an isolated pooled process in which other applications are also run (medium), or in an isolated process separate from other processes (high) [6].
3. Connection Timeout: Sets the length of time in seconds before the server disconnects an inactive user [7].
4. HTTP KeepAlive: Whether to allow a client to maintain an open connection with the Web server [7].
5. ListenBacklog: Set the maximum active connections held in the IIS queue [8].
6. Logging Location: Sets a specific disk and path where the log files are to be saved [9].
7. MaxCachedFileSize: Sets the size of the largest file that IIS will cache [9].
8. MaxPoolThreads: Sets the number of I/O worker threads to create per processor [8].
9. MemCacheSize: Sets the size of the virtual memory that IIS uses to cache static files [9].
10. Number of Connections: Sets the maximum number of simultaneous connections to the site [5].

Figure -2 : Three Tier Architecture



11. Performance Tuning Level: Sets the performance optimization level of IIS to the expected total number of accesses to the Web site per day [7].
12. Resource Indexing: Whether to allow Microsoft Indexing Service to index a specific Web directory and files in that directory [6].
13. worker.ajp13.cachesize: This is not an IIS parameter but it is a configurable parameter of the Web server. It specifies the maximum number of sockets that can be opened between two Tomcat out-of-process processes [10].
14. acceptCount: Sets the maximum queue length for incoming connection requests when all possible request processing threads are in use [10].
15. minProcessors: Specifies the number of request processing threads that are created when a Tomcat connector is first started [10].
16. maxProcessors: Specifies the maximum number of request processing threads to be created by a Tomcat connector, which determines the maximum number of simultaneous requests that can be handled [10].

17. Cursor Threshold: Tells SQL Server whether to execute all cursors synchronously, or asynchronously [9].
18. Fill Factor: Sets the default fill factor for indexes when they are built [9].
19. Locks: Sets the amount of memory reserved for database locks [11].
20. Max Server Memory: Sets the maximum amount of memory, in MB, that can be allocated by SQL Server to the memory pool [11].
21. Max Worker Threads: Determines how many worker threads are made available to the SQL Server process from the operating system [9].
22. Min Memory Per Query: Sets the amount of physical memory in KB that SQL Server allocates to a query [9].
23. Min Server Memory: Sets the minimum, in MB, to be allocated to the SQL Server memory pool [11].
24. Network Packet Size: Sets the packet size that SQL Server uses to communicate to its clients over a network [9].
25. Priority Boost: Whether to allow SQL Server to take on higher priority than other application processes in terms of receiving CPU cycles [9].
26. Recovery Interval: Defines the maximum time, in minutes, that it will take SQL Server to recover in the event of a failure [11].
27. Set Working Set Size: Specifies that the memory that SQL Server has allocated cannot be paged out for another application's use [11].
28. User Connections: Defines the maximum number of concurrent user connections allowed to SQL Server [11].

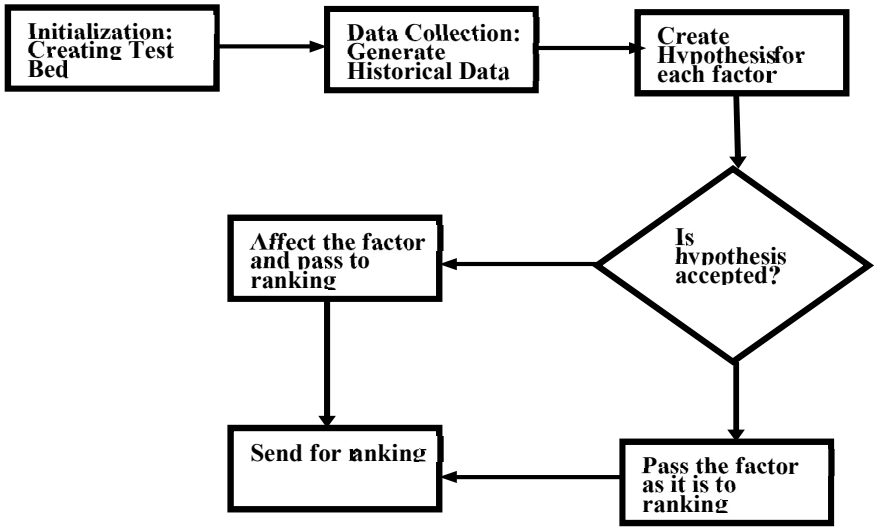
Thus, the optimization of the performance of an e-commerce system is a deal to satisfy all the above twenty-eight factors to their maximum possibility. This makes the process of e-commerce system performance optimization a complex task.

There are numerous methodologies and techniques to maximize every factor individually. But, when it comes to the overall performance of the system, selecting factors to be considered, prioritizing chosen factors, identifying appropriate strategies for individual factors, and creating a package of process that achieves optimal performance is a challenging task.

3. SELECTION OF SIGNIFICANT IMPACT FACTORS

The method for identifying factors that have significant impact on system performance and for ranking them according to their degree of impact consists of three major phases as shown in Figure 3.

Figure 3 : Performance Evaluation Model



Step 1 Initialization: This involves running experiments on all factors in order to determine the "best" initial level of each factor. This is necessary since it would be counterproductive to perform studies on an un-optimized system that may exhibit inferior performance. The details of the initialization process are discussed in the following section.

Step 2 Data Collection: The initialization process yields a system that is well configured to a certain point and exhibits reasonable performance under various levels of workload intensity. The goal of this phase is to determine the factors that are statistically significant and determine the interaction effects, if any, between each factor and the types of requests submitted to the system.

Step 3 Hypothesis Testing and Ranking: After determining the factors that are significant and any interaction effects, the next step is to rank the factors in the order of their impact on various performance metrics. The results obtained in the previous phase are used for ranking purposes.

The purpose of the ranking approach is to sort the various factors in decreasing order of impact on the three metrics of interest: response time, throughput, and probability of rejection. The ranking is constructed by comparing the measured minimum, maximum, and the computed range of the performance metrics (i.e., response times, throughputs, and probabilities of rejection) due to a factor, against the corresponding target performance metrics values.

The original complexity of performance evaluation and optimization arises due to this particular process of ranking. All the existing methods of performance evaluation and optimization use the process of ranking or prioritization of factors that affect performance.

4. FACTORS PRIORIZATION ALGORITHMS

Hypothesis testing and ranking is a very fundamental and pioneer method developed for ranking factors in early [12]. This method identifies the factors which are to be improved from historical data. The higher ranking is given to low performing factors and lower ranking is given to high performing factors as they already doing better. Hypothesis testing and ranking has two different methods of

ranking namely 1) One-way hypothesis testing and 2) Two-way hypothesis testing. The above discussed lower rank for high performing factors and higher rank for low performing factors is based on the method of two-hypothesis testing and ranking [12]. One-way hypothesis ranking simply ignores high performing factors and consider only low performing factors and rank them based on historic data.

Strong quality service enforcement (QoS) algorithm [13] uses heuristic techniques of Artificial Intelligence to identify weakly performing factors. QoS simply any picks a factor and read its present performance level, if it is found to be improved, then QoS employs the optimization process pre-programmed for the particular factors. Thus, QoS uses pre-programmed optimization procedure for every single factor and applies them when and if necessary. The disadvantage of this algorithm is that it simply ignores all the other factors when attempting to improve performance of the chosen factor. As the factors impacting performance of system are closely inter-related, improving performance a factor adversely destroy the performance another factor inter-linked to the previous one.

Markov chain modeling was proposed [14] for ranking or prioritizing the factor. Initially, Markov Chain algorithm generates queue of all variable with present state of performance in descending order. This algorithm lively evaluates the state of each and every factor and compares them. The factors which are performing lower are brought into front part of the queue and the factors which are doing better are brought into rear part of the queue.

Though, Markov Chain Algorithm is found to be more efficient than Strong Quality Enforcement Algorithm and Hypothesis Testing and ranking, it is added load to the actual system. As enormous processor time are needed to read present state of all twenty-eight factors and to organize into rear and front part dynamically during the execution of system, it is found to be vague algorithm [13].

Fixed Rule Based Algorithm was introduced by AT&T [15] which adapted Markov Chain Algorithm but without its dynamic nature. The present state of the factors were periodically tested, say once in a week or month, and the queue is reorganized for once in every week or month bringing low performed factor into front part of queue. Though, this fixed rule Based Algorithm was more successful for smaller systems, it failed utterly at large systems when comparing to other methods.

Swarm Particle Optimization method of Artificial Intelligence was introduced [16] to give ranking or prioritization to factors that affect performance of ecommerce systems. The swarm particle optimization algorithm dealt the problem of ranking almost similar to Fixed Rule Based algorithm but with a little variation. This algorithm assigns a minimum or maximum threshold value for every factor. The Markov queue will never be altered until all the factors are under their threshold. If one or more factors lose their threshold re-organization in the queue is invoked. Swarm article optimization is found to better than Fixed rule Based Algorithm because of less frequent re-organization of queues, but its overall efficiency in optimizing system performance was poor particularly to large systems.

Hop level flow control Algorithm was introduced [17] as an innovative approach for ranking or prioritization factors that affect performance. During the design stage, a default Markov queue is set up and the system is launched with a facility that every user will be asked for feedback suggesting the worst-performing factor using a questionnaire. The worst performing factor indexed by the user will be moved to the very front part of the tree. This questionnaire process is carried out for few numbers of days or months or until a stable performance of the system is achieved. This hop

level flow control algorithm is found to be highly efficient when very few factors are underperforming. If there are many factors that are performing lower, then the algorithm consumes a lot of time to come to a stable state.

5. ANALYSIS OF PRIORIZATION ALGORITHMS

The fundamental modeling of Hypothesis Testing and Ranking appears to be effective if only if a plenty of time is allowed for collecting test data [12]. It is estimated that it has to run for months for medium to large scale systems to give enough test data so that hypothesis can be formulated. In this rapidly growing ecommerce market, hypothesis Testing model is inefficient as it consumes a lot time.

As the factors that affect the performance of systems are closely inter-related improving performance of a factor adversely degrades actually well performing another factor [13]. Thus, an overall system performance is not guaranteed in QoS algorithm.

Whilst, an e-commerce system itself is a complex system with many and many components such as Product listing, product description, online sales system with payment gateways, inventory system, and item delivery tracking system, Markov Chain approach adds more complexity and load to the system though it more efficient than QoS and Hypothesis Testing and Ranking. The variants of Markov Chain Algorithm such as Swarm Particle optimization and fixed rule Based algorithms are found to be more suitable only for smaller systems [15, 16].

The Hop level flow algorithm is efficient only if low performing factors are less in number. When more than the quarter of factors is low performing in a given situation, the hop level flow algorithms fails [17].

When understanding existing methods of performance evaluation and optimization of e-commerce systems, the present study sees a lot of study needed in the area to bring out a perfect and exact algorithm to optimize the performance of e-commerce systems to their best. Also, it is noticeable that all the algorithms analyzed consider the twenty eight one after another using the queue structure. So, the present study proposes that there is a need of an algorithm that considers all the factors simultaneously giving equal importance to all factors at a time, unlike one-after-other queuing model in existing techniques.

6. CONCLUSIONS

Twenty-eight performance impact factors of three tire architecture ecommerce systems, evaluation modeling of e-commerce systems and six different performance impact factors ranking algorithms are studied.

The efficiencies of different ranking algorithms are analyzed and found that either they are inefficient at large systems, or time consuming when considering large number of factors or ignorant to one factor while considering another.

Thus, the present study proposes that there is a need of an efficient algorithm that can overcome above issues and consider multiple factors giving equal importance at a time.

7. REFERENCES

7.1. ***Journal Article***

[1] *S. Vekkataraman, W.K. Fuchs, and J.H. Patel, "Diagnostic Simulation of Sequential Circuits Using Fault Sampling," Proc. IEEE 11th Int'l Conf. VLSI Design: VLSI for Signal Processing, Jan. 1998.*

[2] *L. Hohwiller and S. Wending, "Fieldbus Network Simulation using a Time Extended Estelle Formalism," Proc. IEEE Eighth Int'l Symp. Modeling, Analysis, and Simulation of Computer and Telecomm. System, Aug. 2000.*

[4] *A.R. Larzelere II, "Creating Simulation Capabilities," IEEE Computer Science and Eng., pp. 27-35, Jan. 1998.*

[12] *Avritzer, A., & Weyuker, E. J. (2004). The role of modeling in the performance testing of e-commerce applications. IEEE Transactions on Software Engineering, 30(12), 1072–1083. doi:10.1109/tse.2004.107.*

[13] *A. Avritzer and E.J. Weyuker, "Quality of Service Enforcement for Distributed Objects," IEEE/Proc. Software, vol. 146, no. 5, pp. 232- 244, Oct. 1999.*

[14] *A. Avritzer and E.J. Weyuker, "The Automatic Generation of Load Test Suites and the Assessment of the Resulting Software," IEEE Trans. Software Eng., vol. 21, no. 9, pp. 705-716, Sept. 1995.*

[18] *D. Saravana Prakash et al., "A Survey of Performance Optimization techniques for Web Applications", Gorteria Journal, Vol. 34, Issue 9 – 20211, pp 13-21 (2021).*

[19] *Srivastava, Nishi, Ujjwal Kumar, and Pawan Singh. "Software and Performance Testing Tools." Journal of Informatics Electrical and Electronics Engineering 2.01 (2021): 1-12.*

7.2. ***Book***

[5] *M. Baute, "White Paper: Multi-Tier Performance Tuning," www.gasullivan.com/whitepapers/PerformanceTuning A.pdf.*

[6] *Microsoft, "Windows 2000 Performance Tuning, White Paper," Mar. 22, 2000,http://www.microsoft.com/windows2000/ Professional/ evaluation/performance/reports/perfume.asp.*

[7] *"IIS 5.0 Documentation," Microsoft Windows 2000 Help.*

[8] *B. Curry et al, "The Art and Science of Web Server Tuning with Internet Information Services 5.0," www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol iis/iis5/maintain/optimize/iis5tune.asp.*

[9] *R. Jain, The Art of Computer Systems Performance Analysis, John Wiley & Sons, 1991.*

[10] *A. Bakore et al., "Professional Apache Tomcat," Wrox Press Inc, 2002.*

[11] *E. Whalen et al, "SQL Server 2000 Performance Tuning: Technical*

Reference," Microsoft Press, 2001.

[15]. *B. Curry et al, "The Art and Science of Web Server Tuning with Internet Information Services 5.0. www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnoiis/iis5/maintain/optimize/iis5tune.asp.*

7.3. ***Chapter in a Book***

[17] *M. Gerla and L. Kleinrock, "Flow Control Protocols," Computer Network Architectures and Protocols, Paul Green ed., pp. 361-411, 1982.*

7.4. ***Conference Proceedings***

[3] *A. Avritzer, J. Ros, and E.J. Weyuker, "Estimating the CPU Utilization of a Rule-Based System," Proc. ACM Fourth Int'l Workshop Software and Performance, pp. 1-12, Jan. 2004.*

[16] *V. Cortlelessa and A. Pompei, "Towards a UML Profile for QoS: A Contribution in the Reliability Domain," Proc. ACM Fourth Int'l Workshop Software and Performance, pp. 197-206, Jan. 2004.*