Evaluating the QOS for E-Commerce Architecture Using Proposed WQM Algorithm

Riktesh Srivastava
Information Systems, Skyline University College, University City of Sharjah, Sharjah, UAE

Keywords:

QoS
WQM
Classifier Algorithm
Priority Queue Algorithm Final Queue Algorithm
M/M/n/K
G/G/n/K

ABSTRACT

Impulsive demands posed by unexpected flows of requests from users on Internet have made it exceedingly challenging for E-Commerce to offer anticipated Quality of Service (QoS). In order to provide stable service, it is important for electronic commerce architecture to recognize the QoS (Quality of Service), such as, the response time for requests, delay in accessing the services offered by servers implemented in E-Commerce architecture, and the number of requests waiting in the queue and thereby identifying the strategies to service the requests promptly. The correct and swift QoS can prominently aid to build and operate electronic commerce architecture more proficiently.

In this paper, design and implementation of a QoS-enabled WQM (Web Queue Model) Algorithm is proposed, which uses the queuing models to prioritize the requests from different classes of users. The proposed algorithm takes the requests from clients, and uses three different set of algorithms simultaneously, namely, Classifier Algorithm, Priority Queue Algorithm and Final Queue Algorithm, to improve the QoS for requests. The comprehensive performance of the WQM Algorithm is depicted using various Queuing models and the results are shown accordingly. The queue models which are implemented using the simulation study include the comparative study of the WQM models using M/M/n/K and G/G/n/K models.

Corresponding Author:

Riktesh Srivastava,
Information Systems, Skyline University College, University City of Sharjah, Sharjah, UAE
riktesh.srivastava@gmail.com/rsrivastava@skylineuniversity.ac.ae

1. INTRODUCTION

The Quality of Service (QoS) for any electronic commerce applications plays an essential part in appealing and holding consumers. The workload experienced by these applications tends to vary in a very dynamic way. The complication of the Electronic Commerce architecture shared with the huge short-terms deviations of the workload calls for QoS study for EC configuration. This paper introduces Web Queue Model (WQM), to evaluate the work load of the n-Tier EC architecture implemented in Hybrid Cloud to dynamically monitor and tune so that preferred QoS levels are accomplished. The complete WQM algorithm implemented using Java programming is actually combination of Classifier Algorithm, Priority Queue Algorithm and Final Queue Algorithm. These algorithms execute in concurrency for every requests being received and defines the class of queue. By doing so, different classes of request are generated, which forms the base of WQM algorithm.

The study of web request classification is quite different from text classification, as mentioned in [1]-[7]. These differentiations are:

1) Text classifications are typically performed on structured corpora with well-controlled authoring styles [8], whereas web request classification does not have such property.
2) Secondly, web requests are semi-structured documents, which may have embedded information for depiction; such markup is typically not required for web request classification algorithms.
3) Finally, web requests always exist within a hypertext, whereas, this property is not present in typical text classification problems.

The Classifier algorithm thus designed is based on classification of customers' requests rather that the text classification of customer text. The classification algorithm designed followed the functional point analysis technique. Priority Queue Algorithm is combination of providing the web requests to the appropriate queue and then allocating the Data Center for each class of queue. Final Queue Algorithm is implementation of processing the web requests using either M/M/n/K or G/G/n/K algorithm. Final Queue Algorithm founds its base on the study performed by [9]-[12].

2. ARCHITECTURE FOR WQM IMPLEMENTATION

The complete architecture of the Electronic Commerce architecture for implementing the proposed WQM algorithm includes 4 layers:

1) Web Request Management Server (WRMS)
2) Three Data Centers (DC) with 5 Virtual Machines (VM)
3) Application Servers(AS)
4) Database Servers (DS)

The network diagram demonstrated below intricate the structure for any Electronic Commerce architecture for Hybrid cloud. The point worth noting here is that Web Request Management Server and Data Centers are implemented in the private cloud whereas Application Servers and Data Base Servers are implemented in the public cloud. The Web Request Management Server is the Web Server, implemented using Apache Tomcat, whereas, Data center includes VMware.

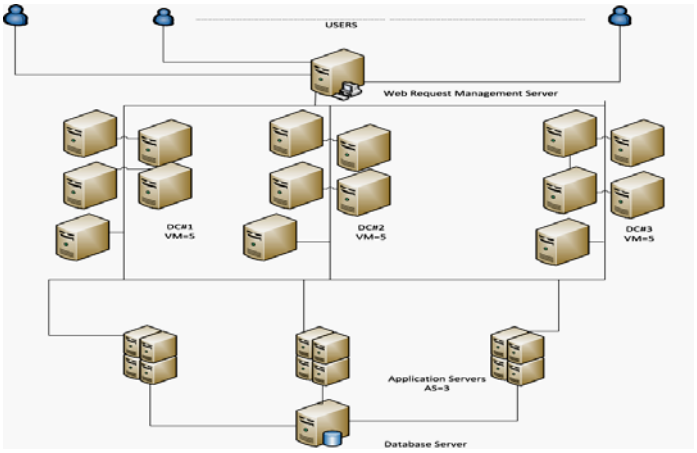


Figure 1: E-Commerce architecture for WQM Algorithm

2.1. Classifier Algorithm

Classifier Algorithm operates at WRMS level with the aim to identify the request type and then classify them into three classes, namely, CDC1, CDC2, and CDC3. Each of these classes has to be processed by any one of the Data Centers with 5 Virtual Machines each. The classification of the request is based on grain size. In WQM algorithm, the three grain sizes are:

1) Fine Grain: If the request to be processed takes less than 5 seconds, based on the request type.
2) Medium Grain: If the request to be processed takes between 5 to 10 seconds, based on the request type.
3) Coarse Grain: If the request to be processed takes more than 10 seconds, based on the request type.

2.2 Priority Queue Algorithm

The Priority Queue Algorithm is designed using Weighted Round Robin (WRR) Technique. In WRR queue algorithm, the requests are categorized into different classes (CDC1, CDC2, CDC3) and then each queue are assigned to the specific queue. The algorithm also addresses the problem of starvation of each request to the final queue algorithm to be processed. The priority queue algorithm is mentioned in Figure 2 mentioned below:

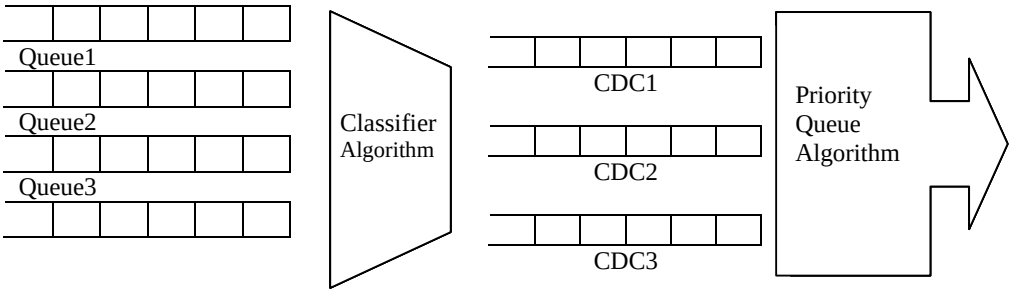


Figure 2. Priority Queue Algorithm Specification

2.3 Final Queue Algorithm

The main experimentation for the proposed WQM algorithm is the final queue algorithm. After the queue is formed to be processed, all the requests must be organized in the final queue. The aim of the final queue algorithm is that each request must be given significance to be processed. Amongst the various queue models available, the analytical study was the comparative study between M/M/n/K and G/G/n/K queue models. The section illustrates the pseudo code of both the queue.

2.3.1 Pseudo code for M/M/n/K Queue Model

```
begin
//Computation of Arrival Sequence
using M/M/n/K Queue Model
read b, SUM=0
read μ, Q(L)
for i=1 to N in step of 1 do
x(i) = RAND U((i))
y(i) = -1/b(log(1-x(i)))
Write y(i)
for i=1 to N in step of 1 do
SUM=SUM+y(i)
Write SUM                <Arrival
Sequence>
end for
end for
//Computation of Departure Sequence
using M/M/n/K Queue Model
for i=1 to N in step of 1 do
x(i) = RAND U(i)
y(i) = -1/b(log(1-x(i)))
Write y(i)
for i=1 to N in step of 1 do
SUM=SUM+y(i)
Write SUM          <Departure
Sequence>
end for
end for
//Computation of Response time for
M/M/n/K Queue Model
for i = 1 to 10 in step of 1 do
Read a(i)
Write a(i)
for i = 1 to 10 in step of 1 do
Read d(i)
Write d(i)
end for
end for
Q=0
for i=1 to 10 in step of 1 do
```

2.3.2 Pseudo code for G/G/n/K Queue Model

```
read n, λ,μ,b,a
read SUM=0
read str
//Generate Gaussian Series
begin
for i=1 to n in step of one do
for j=1 to n in step of one do
sum=0
P(j)=RAND U(j)
sum=sum+P(j)
y(i) = 1/12*sum
Write(y(i))
end for
end for
Y[GAUSSIAN]
//Generate Equiprobable Series
for i=1 to 10000 in step of one do
Generate RAND U(i)
Write((U(i))
end for
Y[EQUIPROBABLE]
//Generate Negative-Exponential
Series
for i=1 to 10000 in step of one do
x(i) = RAND U((i))
y(i) = -1/b log(1-x(i))

Write(y(i))
end for
Y[NEG-EXPONENTIAL]
//Generate Modified-Geometric
Series
b=1/λ
a=1-b
delta=b/a
for i=1 to n in step of one do
x(i)=RAND U((i))
y(i)=(1-exp(delta*x(i))/b
Write(y(i))
```

```
count q(i)
Q=Q+q(i)
Avg Q(L) = Q/N
Write Q(L)
end for
R(t) = 1/μ.Q(L)
Write R(t)
end
```

with the condition

```
//Arr(i) < Dpr(i); i∈(1,N
str= Y[GAUSSIAN] U Y[EQUIPROBABLE]
U Y[NEG-EXPONENTIAL] U Y[MOD-
GEOMETRIC] U Y[BERNAULLI] {D(i,
i∈0,n)}
for i=1 to n in step of one do
Dp(i)=0
Dp(i)=Dp(i) + D(i)
end for
Theta=SUM of Dp(N)
for k=1 to n in step of one do
Dpp(k)=[Dp(k) * N *μ]/Theta −1.0
Write Dpp(k)
end for
//Computation of Response time for
G/G/n/K Queue Model
Read Arrival Instances Arr(i)
having Genenral Distribution
Read Departure Instances Dpp(i)
having Genenral Distribution
for i=1 to n in step of one do
q(i)=[No of arrival Arr(j)<D(j)] −
[i-1]
SUM=SUM+q(i)
end for
for i=1 to n in step of one do
Rt(i)=SUM/n
Write Rt(i)
end for
end
```

```
end for
Y[MOD-GEOMETRIC]
//Generate Bernaulli Series
b=1/λ
a=1-b
for i=1 to n in step of one do
x(i)=RAND U((i))
y(i)=(-a+SQRT(a*a+2*b*x(i))/b
Write(y(i))
end for
Y[BERNAULLI]

// Generation of Departure Sequence
```

3. TECHNICAL SPECIFICATIONS OF THE ELECTRONIC COMMERCE SETUP

Web Request Management Server and three data centers forms the base of the complete experimental setup. For the experiment, Apache Tomcat 7 is used for Web Request Management Server. The challenging job for the study was to setup three Data Centers for the information flow. The specification for each of the Data Centers is:

Data Center 1	Data Center 2	Data Center 3
ID: 0	ID: 1	ID: 2
Number of Processors: 17	Number of Processors: 12	Number of Processors: 10
VM Policy: TIME_SHARED	VM Policy: TIME_SHARED	VM Policy: TIME_SHARED

The public cloud (Data Center 1) was used to set up Application and Data Base Server and was based in US. The other two private clouds (Data Center 2 and Data Center 3) are based in Sharjah within the University premises. There are 5 VM used in two Data Centers 2 and 3, which handles 300 requests per minute and buffer size is 1000 Bytes for each request.

3.1 Load Balancing Policy for the Data Center 2 and Data Center 3

Data Center 2 followed Round Robin technique, wherein, the controller allocates the requests to a list of VMs on a turning basis. The first request is assigned to a VM- selected randomly from the group and then the controller disperses the successive requests in a circular order. Once the VM is consigned the request, the VM is moved to the end of the list. In this technique; there is a better allocation concept known as Weighted Round Robin Allocation in which controller assigns a weight to each VM so that if one VM is capable of handling twice as much load as the other, the powerful server gets a weight of 2. In such cases, the controller assigns two requests to the powerful VM for each request assigned to a weaker one. The major issue in this allocation is this that it does not consider the advanced load balancing requirements such as processing times for each individual requests. Thus the choice is to use RRLB technique for Data Center 2.

Data center 3 uses Throttled Load Balancer (TLB) allocation policy. The TLB preserves a record of the state of each virtual machine (busy/ideal). If a request arrived regarding the allocation of virtual machine, the TLB sends the ID of ideal virtual machine to the data center controller and data center controller allocates the ideal virtual machine.

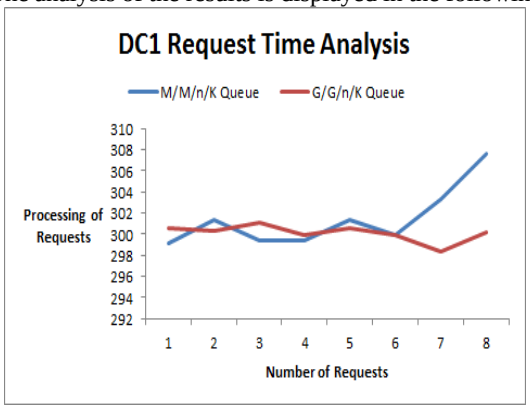
4. RESULTS AND ANALYSIS

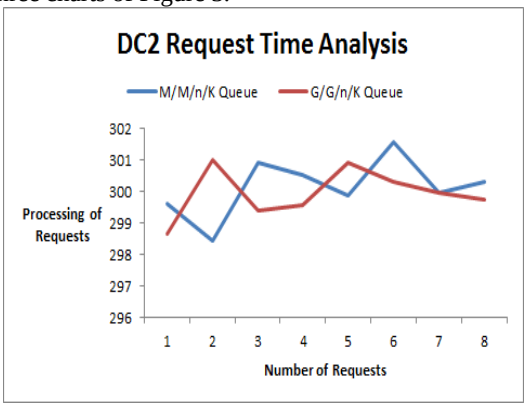
Table 1 demonstrates the experiment outcomes for each of the Data centers w.r.t the number of requests arriving per minute. The table is illustrative in nature and is self-explanatory.

Table 1: Experiment Outcomes

Number of Requests arriving per minute	M/M/n/K Queuing Model			G/G/n/K Queuing Model		
	Processing time at DC1 /minute	Processing time at DC2/minute	Processing time at DC3/minute	Processing time at DC1 /minute	Processing time at DC2/minute	Processing time at DC3/minute
250	299.143	299.614	297.612	300.619	298.637	296.897
300	301.316	298.424	299.781	300.337	300.99	300.012
350	299.391	300.918	301.521	301.09	299.364	297.905
400	299.366	300.509	300.667	299.955	299.536	300.545
450	301.408	299.85	299.984	300.601	300.89	298.997
500	299.962	301.545	301.666	299.957	300.297	300.001
550	303.333	299.937	302.351	298.422	299.965	300.231
600	307.562	300.281	298.983	300.197	299.742	296.887

The analysis of the results is displayed in the following three charts of Figure 3.





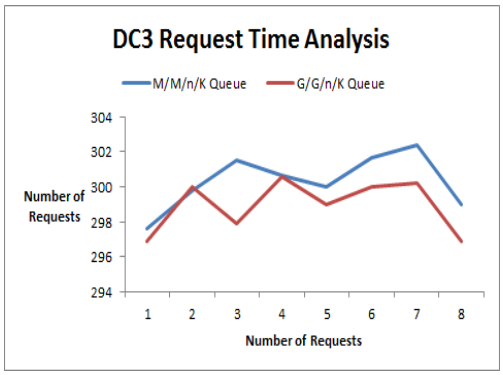


Figure 3: Outcomes of the Experiments

5. CONCLUSION

Based on the outcomes cited in section 4 of the research paper, for DC1 and DC2, the response time is practically the same for the lower number of requests received at the web request management server. However, G/G/n/K model was found to be more appropriate for greater quantity of requests arriving at the architecture. For DC3, both the models were found correspondingly apposite for the number of requests arriving. As a commendation of WQM algorithm, the employment of WQM algorithm using G/G/n/K was found to be more seemly under the given conditions.

REFERENCES

[1] F. Sebastian, "A tutorial on automated text categorisation " , In Proceedings of 1st Argentinean Symposium on Artificial Intelligence (ASAI-99), pp. 7-35, 1999.

[2] F. Sebastiani, "Machine learning in automated text categorization", ACM Computing Surveys 34(1), pp. 1-47, 2012.

[3] K. Aas and L. Eikvil, " Text categorisation: A survey." Technical report, Norwegian Computing Center, P.B. 114 Blindern, N-0314, Oslo, Norway. Technical Report 941, 1999.

[4] A.H. Tan, "Text mining: The state of the art and the challenges " In Proceedings of PAKDD Workshop on Knowledge Discoverery from Advanced Databases, pp. 65–70., 1999.

[5] S. Tong and D. Koller., "Support vector machine active learning with applications to text classification", Journal of Machine Learning Research, pp. 45-66, 2001.

[6] A. Cardoso-Cachopo and A. L. Oliveira., "An empirical comparison of text categorization methods", In Proceedings of the 10th International Symposium on String Processing and Information Retrieval (SPIRE), Volume 2857 of LNCS, Berlin, pp. 183-196, Springer, 2003.

[7] P.N. Bennett, *et. al*., "The combination of text classifiers using reliability indicators", Information Retrieval 8(1), pp.67–100., 2005.

[8] C. M. Chekuri, *et. al*., "Web search using automated classification", In Proceedings of the Sixth International World Wide Web Conference, Santa Clara, CA. Poster POS725, 1997.

[9] R. Srivastava, "Memory Estimation of Internet Server via M/G/1, G/M/1, G/G/1 Queuing Models: A Comparative Perspective ",6th Annual ISOne World Conference, Emerging Academia and Enterprise Agendas, ISOne World, Las Vegas,USA., 2007.

[10] R. Srivastava, "Design and Implementation of G/G/1 Queuing Model Algorithm w.r.t. it's a applicability for Internet Gateway Server, ",International Arab Journal of Information Technology (IAJIT), pp.367-374, 2007.

[11] R. Srivastava , "Estimation of Web Proxy Server Cache Size using G/G/1 Queuing Model", International Journal of Applied Research on Information Technology and Computing (IJARITC). pp.46-58, 2010.

[12] R. Srivastava, "Analysis Of Job Scheduling Algorithm For An E-Business Model In A Cloud Computing Environment Via GI/G/3/N/K Queuing Model", International Journal of Advancements in Technology, pp. 215-229, 2012.

BIBLIOGRAPHY OF AUTHOR

Dr Riktesh Srivastava is presently working as Assistant Professor, Information Systems in Skyline University College, Sharjah, UAE. Dr Srivastava has published more than 50 papers in various International Journals of Computer Science and Information System and is on editorial board of more than 10 International Journals of repute.
His research areas include Distributed Computing, Cloud Computing and Mobile programming.