Evaluation of Response Time Using Gang Scheduling Algorithm for B2C Electronic Commerce Architecture Implemented in Cloud Computing Environment by Queuing Models

Riktesh Srivastava

***Abstract*—Cloud Computing refers to the concept of outsourcing the services, computational requirements and data storage to a centralized facility called "Cloud". Cloud consists of an assemblage of virtualized resources, which include both computational and storage services that can be provisioned on demand, depending on the users' necessities. Gang Scheduling is one of the most efficient algorithms for scheduling parallel jobs and is already functional in the extents of parallel and distributed systems. This paper associates the enactment of gang job scheduling algorithm in B2C electronic commerce architecture to calculate the total response time of the B2C EC architecture in public cloud, using combination of different queuing models. Results reveal the performance of job scheduling algorithm using each of the queuing models and suggestions are accomplished consequently.**

***Index Terms*—M/G/1, G/M/1, M/M/2, M/M/m G/G/1, cloud computing, response time, gang scheduling algorithm.**

I. Introduction

In a cloud computing environment, there is mounting demand for good job scheduling algorithm that will accomplish the jobs in a subcontracted environment. However, it is problematic to schedule the tasks on distributed systems, like cloud computing, among innumerable competing jobs from varied sources. The complication arises as the cloud computing environment consists of several, loosely interconnected processors, where jobs to be processed by different servers and various techniques are to be used to coordinate processing. To determine this,it is critical to appropriately assign the tasks to servers and then schedule execution on distributed servers to proficiently schedule parallel jobs. Good scheduling policies can maximize overall performance of the B2C EC architecture and avoid unnecessary delays. One idea is gang scheduling algorithm, where a set of jobs are scheduled to execute simultaneously on the set of servers.

In this study there are several requests which consist of parallel tasks that are scheduled to accomplish concurrently on a set of servers implemented for B2C EC implemented in a Cloud Computing environment. These numerous requests arrived at the web server from numerous sources need to start essentially at the same time, co-ordinate their executions and compute at the same pace. This entire process is managed through the "gang scheduling algorithm" using "queuing theory" which allows requests to interact efficiently by using busy waiting, without the risk of waiting for a task that is not currently running.

Because gang scheduling stresses that no request complete unless all other gang member requests execute, some servers may remain idle even when there are requests waiting to be run. With gang scheduling, at any time there is a one-to-one mapping between requests and servers. Although the total number of requests in the system may be larger than the number of servers, no gang contains more requests than it does servers. We assume that all the tasks within the same gang execute for the same amount of time, i.e, .the computational load is balanced between them.
Gang scheduling in distributed and parallel systems has been studied by many authors, such as [1]-[4], The concept of queuing theory and its implementation is studied in [5]-[8].

The structure of this paper is as follows. Section 2 introduces the model of implementation of B2C electronic commerce architecture in a Cloud computing environment. Section 3 presents the metrics used to evaluate the performance of the scheduling policies. Section 4 displays the simulation results are both presented and analyzed. Finally, section 5 summarizes the paper and provides recommendations for further research.

II. B2C Electronic Commerce Architecture and Its Implementation in Public Cloud

The proposed B2C EC architecture is an extension to the system of Client Server Computing. In the architecture, the Application Server and the Database Server are implemented in the public cloud. So the customer using the application is not fretful with the complexities of the business logic and is offered the complete web application with added service. Thus, the architecture offers a significant workload shift.

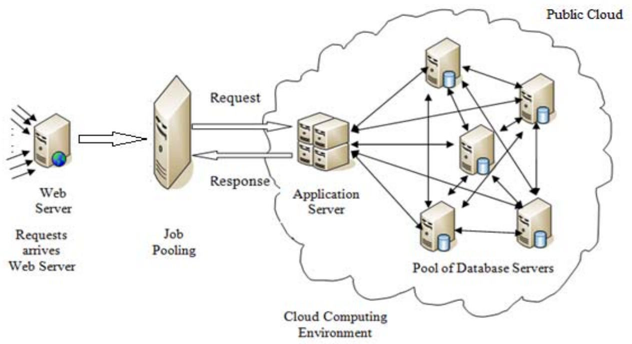

Fig. 1. B2C EC Architecture implemented in public cloud

Manuscript received September 10, 2012; revised October 20, 2012.
Riktesh Srivastava is with Skyline University College Information Systems, UAE (e-mail: riktesh.srivastava@ gmail.com/ rsrivastava@skylineuniversity.com).

In the proposed architecture, the Web server has to no longer do the entire profound lifting when it comes to running applications. The network of clouds, which includes Application Server and Database Server(s), hold the impediments of the architecture. Also, the hardware and software demands on the user's side dwindle and the web server only executes the architectures interface software, whilst the cloud network takes care of the rest. The comprehensive architecture is depicted in Fig.1

A. Step by Step Architecture Explanation

As illustrated in Fig. 1, all the clients requests is being received at the Web Server. Since, there is an adoption of Markovian distribution; the Web Server maintains the first queue (Q_1) and transfers the requests to one of the Job Pooling Server. Job Pooling Server which stores the request in a queue (Q_2). The set of requests is then transferred to the Application server. The queue is named as Q_3. Application Server generates the business logic, is implemented in the public cloud. All the set of database requests, being processed by the pool of database servers, is taken from the queue, named, Q_4.

For simplicity, the entire concept is elaborated in the figure Fig. 2, which forms the base for the mathematical analysis of the model.

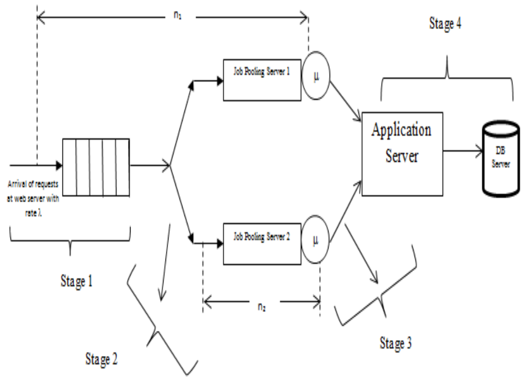

Fig. 2. 4 stages of B2C EC architecture

The selection of queuing models for gang scheduling algorithm is as follows:

Stage 1: A choice between M/G/1, G/M/1 or G/G/1 queuing models

Stage 2: M/M/2 queuing model

Stage 3: M/M/m queuing model

Stage 4: Choice between M/G/1, G/M/1 or G/G/1 queuing models

Based on the Fig. 2, the total response time for the complete model is:

$$R(t) = R(t)_{Stage\ 1} + R(t)_{Stage\ 2} + R(t)_{Stage\ 3} + R(t)_{Stage\ 4} \tag{1}$$

It must be noted that the number of requests at the web server is $\boldsymbol{\lambda}$ and number of responses is $\boldsymbol{\mu}$. For the worst case, $\boldsymbol{\mu}=\boldsymbol{\lambda}+1$.

III. Computation of Response Time at Each Stage

From Fig. 2, response time needs to be calculated for each stage of the B2C EC architecture.

A. Response Time for Stage 1

As already indicated in section 2, for stage 1 we have a choice between M/G/1, G/M/1 or G/G/1 queuing models. The study is already conducted by [5], which is given in Table I:

TABLE I: Response Time for M/G/1, G/M/1 and G/G/1

λ	μ	M/G/1	G/M/1	G/G/1
5000	5001	5102	5098	5538
7500	7501	7653	7647	8400
10000	10001	10204	10194	11190
12500	12501	12755	12740	14126
15000	15001	15306	15283	17695

If we plot the graph for these rates of arrivals, it forms the curve of the form:

$$Y_\phi = a_0 + a_1\lambda + a_2\lambda^2 \tag{2}$$

In order to calculate a_0,a_1 and a_2, a non-linear regression technique is used as already studied by [5],[7],[8].

$$\left.\begin{aligned} n_{a_0}+a_1\sum\lambda_i+a_2\sum\lambda_i^2&=\sum\lambda_i \\ a_0\sum\lambda_i+a_1\sum\lambda_i^2+a_2\sum\lambda_i^3&=\sum\lambda_i y_i \\ a_0\sum\lambda_i^2+a_1\sum\lambda_i^3+a_2\sum\lambda_i^4&=\sum\lambda_i^2 y_i \end{aligned}\right\} \tag{3}$$

Based on the response time mentioned in Table II, we get the following values for a_0, a_1 and a_2, wrt to these selected models.

TABLE II: Values for A_0, A_1 and A_2

M/G/1	G/M/1	G/G/1
a_0=1484.0	a_0=1484.5	a_0=1483.5
a_1=0.873200	a_1=0.873201	a_1=0.873377
a_2=0.000005	a_2=0.0000051	a_2=0.000006

From Table II, the response time for the three models is:

$$484.0 + 0.873200\lambda + 0.000005\lambda^2 \tag{4}$$

$$R(t)_{M/G/1} = 1484.0 + 0.873200\lambda + 0.000005\lambda^2 \tag{5}$$

$$R(t)_{G/G/1} = 1483.5 + 0.873377\lambda + 0.000006\lambda^2 \tag{6}$$

B. Response Time for Stage 2

For stage 2, single queue is divided into 2 queues, one each for the two job pooling server. The balance diagram to evaluate the equation is mentioned in Fig. 3

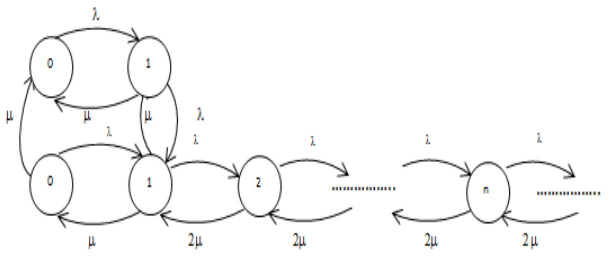

Fig. 3. Balance diagram for M/M/2 queue model

In order to calculate the response time, we need to first estimate the balance equations of Fig. 3. It must be also noted that since there are 2 job pooling servers, to evade any bottleneck in the queue, assume that $\boldsymbol{\mu}_1=\boldsymbol{\mu}_2=\boldsymbol{\mu}$.

$$\lambda p(0,0) = \mu p(1,0) + \mu p(0,1) \tag{7}$$

$$(\lambda + \mu) p(1,0) = \mu p(1,1) + \lambda p(0,0) \tag{8}$$

$$(\lambda + \mu) p(0,1) = \mu p(1,1) \tag{9}$$

$$(\lambda+2\mu)p(1,1) = (2\mu)p(2,1) + \lambda p(0,1) + \lambda p(1,0) \tag{10}$$

$$(\lambda+2\mu)p(n,1) = (2\mu)p(n+1,1) + \lambda p(n-1,1) + \lambda p(n-1,0) \quad \forall n > 1 \tag{11}$$

Traffic intensity for the system is $\delta = \dfrac{\lambda}{2\mu}$

From the equation (11), we get

$$p(n,1) = \frac{\lambda}{2\mu} p(n-1,1) \quad \forall n > 1 \tag{12}$$

Clarifying equation (12), the generic form of the equation can be stated as:

$$p(n,1) = \delta p(n-1,1) = \delta^{n-1} p(1,1) \quad \forall n > 1 \tag{13}$$

From equations (7)-(11), we get be elimination

$$p(0,1) = \frac{\delta}{1+2\lambda}\frac{\lambda}{\mu} p(0,0)$$

$$p(1,0) = \frac{1+\delta}{1+2\delta}\frac{\lambda}{\mu} p(0,0)$$

$$p(1,1) = \frac{\delta}{1+2\lambda}\frac{\lambda(\lambda+\mu)}{\mu^2} p(0,0)$$

Now, observing that

$$\left[\sum_{n\geq 1} p(n,1)\right] + p(0,1) + p(1,0) + p(0,0) = 1$$

Thus, we get

$$\left(\sum_{n\geq 1} p(n,1)\right) p(1,1) + p(0,0)\left[\frac{\delta}{1+2\delta}\frac{\lambda}{\mu} + \frac{1+\delta}{1-\delta}\frac{\lambda}{\mu} + 1\right] = 1$$

or,

$$\frac{1}{1-\delta}\frac{\delta}{1+2\delta}\frac{\lambda(\lambda+\mu)}{\mu^2} p(0,0) + p(0,0)\left[\frac{\delta}{1+2\delta}\frac{\lambda}{\mu} + \frac{1+\delta}{1-\delta}\frac{\lambda}{\mu} + 1\right] = 1$$

From the above-mentioned equation, we get

$$p(0,0) = \left[1 + \frac{\lambda(\lambda+\mu)}{\mu^2(1+2\delta)(1-\delta)}\right]^{-1} \tag{14}$$

The average number of requests at the Job pooling server is computed by observing the number of requests in the system in state (n_1, n_2) is n_1+n_2. Thus, the average number of requests is:

$$= \sum_{k\geq 0} kp(k,0) + \sum_{k\geq 0}(k+1)p(k,1)$$

$$= p(1,0) + p(0,1) + \sum_{k\geq 1}(k+1)p(k,1)$$

$$= p(1,0) + p(0,1) + \sum_{k\geq 1}(k+1) + \sum_{k\geq 1} kp(k,1)$$

$$= 1 - p(0,0) + p(1,1)\sum_{k=1}^{\infty} kp^{k-1}$$

$$= 1 - p(0,0) + \frac{p(1,1)}{(1-\delta)^2}$$

Thus, the average number of requests is

$$\frac{1}{A(1-\delta)^2} \tag{15}$$

where $A = \left[\dfrac{\mu^2(1+2\delta)}{\lambda(\lambda+\mu)} + \dfrac{1}{1-\delta}\right]$

So, the average response time at stage 2 using little's formula is *Average number of requests/$\boldsymbol{\lambda}$.*
Thus,

$$R(t)_{Stage2} = \frac{1}{A(1-\delta)^2.\lambda} \tag{16}$$

Solving equation for the worst case, $\boldsymbol{\mu}=\boldsymbol{\lambda}+1$ and substituting it in equation (16), the final equation for response time at stage is depicted in equation (17).

$$R(t)_{Stage2} = \left[\frac{2\lambda^2+3\lambda+1}{2\lambda^2+4\lambda+1}\right] \tag{17}$$

C. Response Time for Stage 3

As indicated in Fig. 2, at stage 3 the two queues from Job Pooling Servers are merged together, which makes it a typical case of M/M/m queuing model. The generic state diagram for the system is depicted in Fig. 4. It must be noted that in the figure, k illustrates the number of stages, which is 2 in the case of B2C EC architecture.

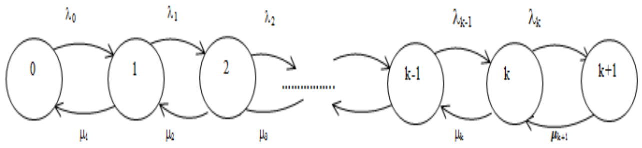

Fig. 4. State diagram for M/M/m queue model

According to [6], the balance equation for M/M/m queue is dependent upon the linear homogenous equation of the form:

$$0 = \sum_{i\neq j} p_i q_{ij} - p_j q_j \tag{18}$$

where p_i and p_j are the transition probabilities wrt the transition rates q_{ij} and q_j.
Using equation (17), following set of equations are observed:

$$0 = -(\lambda_k + \mu_k)p_k + \lambda_{k-1}p_{k-1} + \mu_{k+1}p_{k+1} \tag{19}$$

$$0 = -\lambda_0 p_0 + \mu_1 p_1 \tag{20}$$

Rearranging equation (18),

$\lambda_k p_k - \mu_{k+1} p_{k+1} = \lambda_{k-1} p_{k-1} - \mu_k p_k = \ldots\ldots = \lambda_0 p_0 - \mu_1 p_1$ But from equation (19), $\lambda_0 p_0 - \mu_1 p_1 = 0$. It follows that $\lambda_{k-1} p_{k-1} - \mu_k p_k = 0$. Rearranging,

$$p_k = \frac{\lambda_{k-1}}{\mu_k} p_{k-1} \quad \forall k \geq 1$$

Thus,

$$p_k = \frac{\lambda_0 \lambda_1 \ldots\ldots\ldots \lambda_{k-1}}{\mu_1 \mu_2 \ldots\ldots\ldots \mu_k} p_0 = p_0 \prod_{i=0}^{k-1}\left(\frac{\lambda_i}{\mu_{i+1}}\right) \quad \forall k \geq 1 \tag{21}$$

Using equation (20), the steady state probabilities for the Application Server can be evaluated as

$$p_k = p_0 \prod_{i=0}^{k-1}\left(\frac{\lambda}{(i+1)\mu}\right)$$

$= p_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{k!} \quad \forall k \geq m$ (where m is the number of servers)

$$p_k = p_0 \prod_{i=0}^{m-1} \left(\frac{\lambda}{(i+1)\mu} \right) \prod_{j=m}^{k-1} \frac{\lambda}{m\mu} \tag{22}$$

$$= p_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{m!\,m^{k-m}} \quad \forall k \geq m$$

Defining $\delta = \left(\frac{\lambda}{m\mu}\right)$, the condition for the stability is given by $\delta < 1$. The expression for δ_0 is obtained using equation (21) and the fact that $\sum_{k=0}^{\infty} p_k = 1$.

$$p_0 = \left[\sum_{k=0}^{m-1} \frac{(m\delta)^k}{k!} + \frac{(m\delta)^m}{m!} \frac{1}{1-\delta} \right]^{-1} \tag{23}$$

Thus, the average number of requests at the Application Server is given by the equation (23) as stated below:

$$\sum_{k \geq 0} k p_k = m\delta + \delta \frac{(m\delta)^m}{m!} \frac{p_0}{(1-\delta)^2}$$

$$2\delta + \frac{2(2\delta)^2}{2!} \frac{p_0}{(1-\delta)^2}$$

$$\delta = \frac{\lambda}{2\mu}$$

$$p_0 = \left[1 + 2\delta + \frac{(2\delta)^2}{2!} \frac{1}{1-\delta} \right]^{-1}$$

$$= \frac{1-\delta}{(1-\delta)(1+2\delta) + 2\delta^2} = \frac{1-\delta}{1+\delta}$$

$$2\delta + 2\delta^3 \frac{1-\delta}{(1+\delta)(1-\delta)^2} = \frac{2\delta\left(1-\delta^2+\delta^2\right)}{1-\delta^2} = \frac{2\delta}{1-\delta^2}$$

$$= \frac{4\mu}{4\mu^2 - \lambda^2}$$

$$\mu = \lambda + 1$$

Since, the two queues are merged for the Application Server to generate the business logic, the value of m=2.

$$2\delta + \frac{2(2\delta)^2}{2!} \frac{p_0}{(1-\delta)^2}$$

where $\delta = \frac{\lambda}{2\mu}$

Using equation (22), following value of p_0 is obtained:

$$p_0 = \left[1 + 2\delta + \frac{(2\delta)^2}{2!} \frac{1}{1-\delta} \right]^{-1}$$

$$= \frac{1-\delta}{(1-\delta)(1+2\delta) + 2\delta^2} = \frac{1-\delta}{1+\delta}$$

Thus, the average number of requests at Application Server becomes:

$$2\delta + 2\delta^3 \frac{1-\delta}{(1+\delta)(1-\delta)^2} = \frac{2\delta\left(1-\delta^2+\delta^2\right)}{1-\delta^2} = \frac{2\delta}{1-\delta^2}$$

The average response time at stage 3 using Little's formula is *Average number of requests/λ.*

$$= \frac{4\mu}{4\mu^2 - \lambda^2} \tag{24}$$

Putting $\mu = \lambda + 1$, we obtain

$$R(t)_{Stage3} = \left[\frac{4(\lambda+1)}{3\lambda^2 + 8\lambda + 4} \right] \tag{25}$$

D. Response Time for Stage 4

Response time for stage 4 is identical as specified for stage 1.

IV. Computation of Response Time for Gang Scheduling Algorithm Based on Queuing Models

Based on equation (1) and choice between M/G/1, G/M/1 and G/G/1 queuing models, the response time can be

If M/G/1 model is selected at stage 1 and 4, the total response time would be

$$2\times\left[1484.0 + 0.873200\lambda + 0.000005\lambda^2\right] + \left[\frac{2\lambda^2+3\lambda+1}{2\lambda^2+4\lambda+1}\right] + \left[\frac{4(\lambda+1)}{3\lambda^2+8\lambda+4}\right] \tag{26}$$

If G/M/1 model is selected at stage 1 and 4, the total response time would be

$$2\times\left[1484.5 + 0.873201\lambda + 0.0000051\lambda^2\right] + \left[\frac{2\lambda^2+3\lambda+1}{2\lambda^2+4\lambda+1}\right] + \left[\frac{4(\lambda+1)}{3\lambda^2+8\lambda+4}\right] \tag{27}$$

And if, G/G/1 model is selected at stage 1 and 4, the total response time would be

$$2\times\left[1483.5 + 0.873377\lambda + 0.000006\lambda^2\right] + \left[\frac{2\lambda^2+3\lambda+1}{2\lambda^2+4\lambda+1}\right] + \left[\frac{4(\lambda+1)}{3\lambda^2+8\lambda+4}\right] \tag{28}$$

The experiment conducted using Java using JDK 7.0 shows the simulated results for the three models are given in Table III as under: The representation of the results is depicted in Fig. 5.

TABLE III: COMPUTATION OF RESPONSE TIME W.R.T QUEUING MODELS

Number of requests	Response time –G/M/1 model	Response time –M/G/1 model	Response time –G/G/1 model
10000	21433	21454.02	21635.54
15000	31415	31461.03	31869.31
20000	41897	41978.04	42703.08
25000	52879	53005.05	54136.85
30000	64361	64542.06	66170.62
35000	76343	76589.07	78804.39
40000	88825	89146.08	92038.16
45000	101807	102213.1	105871.9
50000	115289	115790.1	120305.7
55000	129271	129877.1	135339.5
60000	143753	144474.1	150973.2
65000	158735	159581.1	167207
70000	174217	175198.1	184040.8
75000	190199	191325.2	201474.6
80000	206681	207962.2	219508.3
85000	223663	225109.2	238142.1
90000	241145	242766.2	257375.9
95000	259127	260933.2	277209.6
100000	277609	279610.2	297643.4

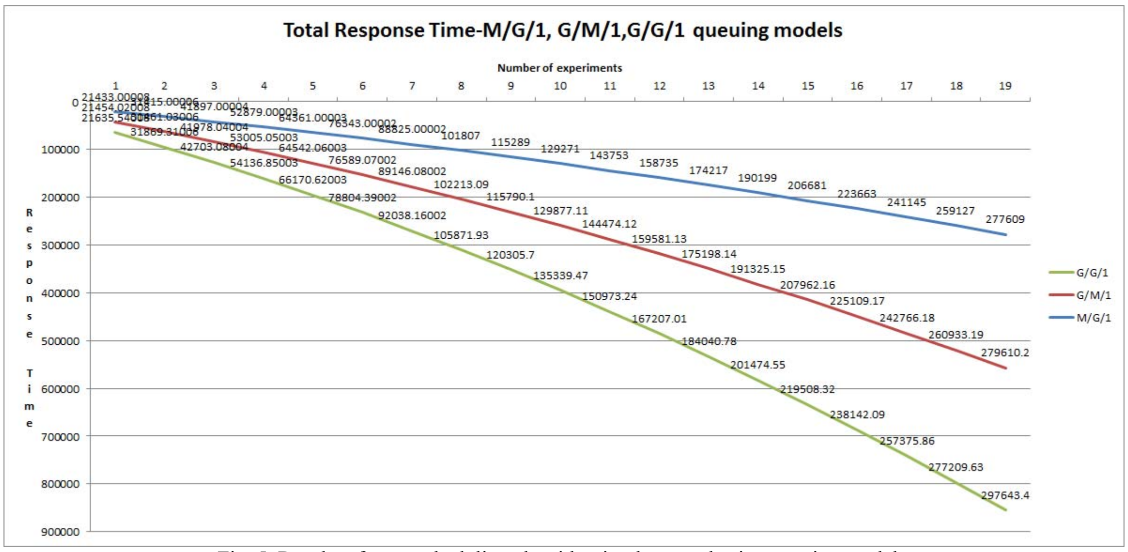

Fig. 5. Results of gang scheduling algorithm implemented using queuing models

V. Conclusions and Recommendations

The study steered recommends that M/G/1 queuing model is best matched for instigating the gang scheduling algorithm in cloud computing environment, followed by G/M/1 and G/G/1 respectively. Simulated results also depicts that there is exact less variance between the response time, when the number of requests are less and the difference upsurges when the requests are more. So, while employing the model for fewer numbers of requests, one can select any one of the models, and when the numbers of requests are more, M/G/1 model is best matched for implementing gang scheduling algorithm in public cloud.

References

[1] K. Aida, “Effect of Job Size Characteristics onJob Scheduling Performance,” *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, Springer-Verlang, Berlin, Germany, vol. 1911. pp. 1-10, 2000.

[2] D. G. Feitelson and L. Rudolph, “Evaluation ofDesign Choices for Gang Scheduling Using Distributed Hierarchical Control,” *Journal of Parallel and Distributed Computing,* Academic Press, New York, USA, vol. 35, pp. 18-34, 1996.

[3] C. Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, and M. Miller, “Rotation, scale, and translation resilient public watermarking for images,” *IEEE Trans. Image Process*, vol. 10, no. 5, pp. 767-782, May 2001.

[4] D. G. Feitelsonand and M. A. Jette, “Improved Utilisation and Responsiveness with Gang Scheduling,” *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, Springer-Verlang, Berlin, Germany, vol. 1291, pp. 238-261.

[5] E. Frachtenberg, D. G. Feitelson, F. Petrini, and J. Fernandez, “Adaptive Parallel Job Scheduling with Flexible Coscheduling,” *IEEE Transactions onParallel and Distributed Systems*, IEEE Computer Society, Los Alamitos, CA, USA, vol. 16, no. 11, pp. 1066-1077.

[6] H. Poor, *An Introduction to Signal Detection and Estimation*, New York: Springer-Verlag, 1985.

[7] L. K. Singh and R. Srivastava, “Memory Estimation of Internet Server using Queuing Theory: Comparative Study between M/G/1, G/M/1 and G/G/1 Queuing Models,” *International Journal of Computer and Information Science and Engineering*, vol. 1, no. 2, pp. 125-129, 2007.

[8] K. S. Trivedi, “Probability and Statistics with Reliability, Queuing and Computer Science Applications,” in *Proc. of 12th Edition*, PHI, pp. 363-364, 2001.

[9] R. Srivastava, “Estimation of Buffer Size of Internet Gateway Server via G/M/1 Queuing Model,” *International Journal of Applied Science, Engineering and Technology*, vol. 19, pp. 474-482, 2007.

[10] R. Srivastava *et al*, “Estimation of Buffer Size of Internet Gateway Server via G/M/1 Queuing Model,” *International Journal of Applied Science, Engineering and Technology*, vol. 4, no. 1, 2007.

Riktesh Srivastava is an Assistant Professor at Skyline University College, Sharjah, UAE. He holds Doctrate degree in Electronics, apart from MTech (IT), MS (Electronics) and MBA. He is authored more than 30 papers for various international journals of repute and is member of editorial board for 10+ international journals of computer science, information systems.