Queuing Management with Feedback in Cloud Computing Centers with Large Numbers of Web Servers

A. Z. Melikov[1(✉)], A. M. Rustamov[2], and J. Sztrik[3]

[1] Institute of Control Systems, ANAS, B. Vahabzade 9, AZ1141 Baku, Azerbaijan
agassi.melikov@gmail.com
[2] Baku Engineering University, H. Aliyev 120, Khirdalan, AZ0101 Baku, Azerbaijan
anrustemov@beu.edu.az
[3] University of Debrecen, Debrecen 4032, Hungary
sztrik.janos@inf.unideb.hu

Abstract. The objective of the paper is to analyzed QoS metric of cloud computing with large-scale of web server from queue management perspective. We propose a new method in order to effectively calculate the steady-state probabilities of cloud system with a large number of web servers. Numerical results showed that the proposed algorithms have higher accuracy and negligible computation time. Taking into account that in the cloud computing repair of server is taking some hours and response time is handling within some seconds, we can correctly apply space-merging algorithm.

Keywords: Queuing system · Cloud computing · Cloud technology

1 Introduction

Last decades, the optimization of applications and backend processing in the computing and virtualization technologies is becoming the integral part of the nowadays Internet applications. The main indicators of these technologies are still remaining the same: (1) high service rate and (2) cost-effectiveness. The main paradigm of the cloud technologies is to share computing resources among application within internal network or in the global network (the Internet) by providing the main two indicators mentioned above. All public cloud environments IaaS (Infrastructure-as-a-Service), PaaS (Platform-as-a-Service), and SaaS (Software-as-a-Service) should be arranged that any computation should be serviced with low resources [1,2].

Technically, cloud computing is set of servers with different specification, switches and routers. And servers are mainly considered as set of the hardware, software, environment, and human factors.

In cloud computing main servers that accepts the request from the users are called web servers. Web servers are one of the main factors that affect QoS metric

V. M. Vishnevskiy and D. V. Kozyrev (Eds.): DCCN 2018, CCIS 919, pp. 106–119, 2018.
https://doi.org/10.1007/978-3-319-99447-5_10

of the each request coming from users. From general point of view, decrease in latency and response times can be caused by *client's attitude, network,* and *server side* in the transaction in the client/server paradigm.

As we see, cloud computing has been deeply investigated by many scholars within the last decade from different perspective, such as network optimization, server utilization, QoS metric optimization, software architecture and etc. Among these directions, QoS metric of the servers (mainly web servers) is also studied by many scholar taking into account different QoS metric parameters.

Investigation shows that end-users are patient for the maximum delay in each request up to eight-ten seconds. That's way most of the researches are concentrating on providing service rate up to maximum eight-ten seconds, in ideal case one-three seconds [3].

Network services and system architecture planning are heavily dependent on the prediction of demand of the requested services. Nowadays, enormous increase of data size (video, picture, sound files and streaming) make scholars to find optimal solution for the serving them in short time within the minimum computation, that is the main constraint of the internet and cloud computing. In that case Poissonian properties of the arrival packets are little bit challenge to be modeled taking into account many attributes. Because of its tremendous analytical qualities, Poisson processes are widely used in network and system architecture planning and analysis. By implementation that kind of solution may result in non-negligible ramifications for the QoS offered, such as network optimization, queue management, platform optimization and etc.

In this paper, we will consider queuing management with feedback in cloud computing centers with large numbers of web servers. Our contribution in this kind research is feasibility of implementation of state merging algorithm because of unit difference in the server repair time and interarrival times of jobs. Namely, server repair time is calculated in hours, where interarrival times of jobs is with seconds.

The rest of the paper is organized as follows. In Sect. 1 detailed literature review about implementation of queuing system in the cloud computing is analyzed. In the Sect. 2 physical model of the proposed queue management in the cloud system is given, and both exact and approximate methods are detailed explained. Sections 3 and 4 are about the numerical results, and conclusion and future works, respectively.

2 Related Work

Modern web servers are extremely distinguished from small server to the giant computing system depending on services they're performing and number of users [4]. Because of network capabilities some arrival packet may be dropped because of no idle place in the processing queue, even though the server is under fairly light load [5]. Erramilli et al. [6] investigated dependence of traffics that required long time of service on queue management system, and found positive correlation between feasibility and practical impact on them. Authors showed that the key parameters of the cloud computing: buffer sizing, admission control and rate control have significant affect on the number of arrival and reserving packets.

In [7] author modeled software for apache web server where it can manage several parallel threads at the same time working independently or dependently with each other. They mentioned that implementation of static queue management system, such as FCFS is not proper for the web servers, further more simple queuing models don't satisfy modern web servers requirement. Today's cloud computing implementation diverse significantly depending on the their application area. Thus web servers should management the queue and serving mechanism so that minimum computation is done. Processes running at once, several threads running once, or with a combination. They also showed that neither service nor arrival processes is necessarily Poissonian. They implemented the M/G/1/K/PS queue management model in their research. The objectives of this study was to investigate the probability of blocking of the server within the given model. They have simulated arrival packets similar to the HTTP packets with Poisson and General distribution function. Congestion in the network and the probability of blocking in the web server side extremely affect the entire performance of the cloud computing. Because the web server as assumed as gateway to the cloud system. Mei et al. [5] deeply analyzed the effects of congestion in networks for response time and the probability of blocking in the web server side and found positive correlation between them.

In [8–13] some other parameters (such as response time, throughput and network utilization) of QoS metric in cloud computing were investigated. In [8], the authors applied a classic M/M/m model in order to get the response time distribution of a cloud system. Inter-arrival and service time were taken by exponential distribution function. In [12], Karlapudi's developed performance tool in order to predict web server demand on different response scenarios. In [13], Mei analyzes QoS metric in VoIP services within the cloud system.

In [14] the authors showed that the servers within the cloud system that utilize 60% of their resources in the peak power, can run on 20–30% utilization. They found PowerNap solution that save the energy about 23% by optimization queue in the system.

In [15], authors touched the same problem mentioned in [14]. They found that in the cloud systems with thousand of servers there is a 7–16% gap between real utilization of resources and suggested by the manufacturer. They also mentioned that each data center within the cloud system must have their own queue and resource management system based the real requirements. QoS metric performance evaluation of cloud technology through the simulation sometimes doesn't give precise result as we expected [16,17].

A cloud computing with multi-server system models many times faces with the server failures. The main reason of these failures comes from the inaccurate and imprecise queueing management model. This factor severely affect the entire performance of the cloud computing [18–27].

Enver [28] analyzed the ability to deliver acceptable levels of QoS for the cloud systems, and requirements for the performance of the system. Author proposed the analytical solution approach in order to solve QoS metric calculation problem within the large-scale systems in cloud computing with the presence

of failures and repairs. It is mentioned that there're significant difference in the final simulation results. Author's solution mainly focused on level decomposition approach.

As we see different models of queuing management system were implemented in order to analyze cloud computing, mainly web servers performance. However, to the best of our knowledge, so far feedback queuing model didn't applied to the cloud system. Similar approach is done by Chakka [29] for large-scale of networks.

In this paper we analyzed QoS metric of cloud computing with large-scale of web server from queue management perspective. Unlike [28] we take into account of being impatient of call when there isn't any operative server in the system. Moreover, we propose a new method in order to effectively calculate the steady-state probabilities of cloud system with a large number of web servers. By implementation of these steady-state probabilities, it's possible to calculate desired QoS metrics with low computation. Numerical results showed that the proposed algorithms have higher accuracy and negligible computation time. Taking into account that in the cloud computing repair of server is taking some hours and response time is handling within some seconds, we can correctly apply space-merging algorithm [30].

3 Exact and Approximate Methods of the Proposed Model

The physical model of the proposed queue management system in the cloud computing is given in Fig. 1. Clients are simultaneously sending requests. When the requests arrive to the web server, they have to wait for a while in the queue. The main point here is that a number of active thread in the web server. By default web server are generation one thread per request, and it results in increase in utilization and memory. Web servers must optimally handle the number of thread and buffer management, so that, CPU utilization and RAM usage should be below some thresholds. In our model request in any case (failure, not accepted, not validated and etc.) should return the client.

Jobs arrive to the servers from different users independently according to Poisson distribution with rate of λ. The service times for each job are independent and identically distributed (i.i.d.) random variable (r.v.) with exponential distribution with the rate μ. There are N number of servers in the cloud system and the total capacity of the system is R, i.e. the buffer size for waiting of jobs is equal to $R - N$. The queue is handled by arbitrary conservative discipline which means that if there are jobs in system then server does not idle. In other words, when a user has a job from the cloud system, in case at least one of the servers are idle, the job will be handled by one of the idle servers; otherwise, if all the servers are busy and the queue is sufficient to accept the incoming job, it will join the queue. Unlike the [14] here it is assumed that jobs in queue are impatient, i.e. every jobs in queue independently other ones waits in the queue

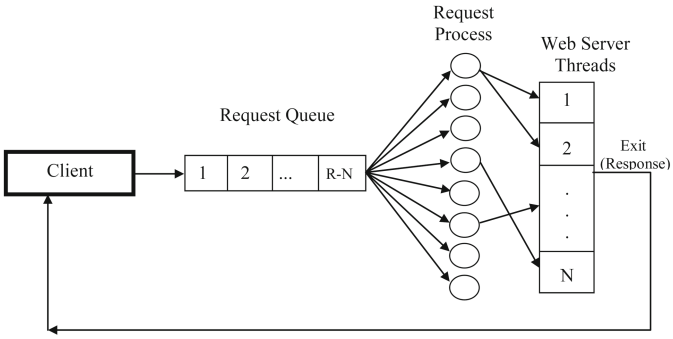

Fig. 1. The physical model of proposed queue management in cloud system

during random time, which has exponential distribution with mean α^{-1}. If the queue is full, the incoming job will be lost (rejected).

The servers considered are prone to failures where operative periods (i.e. period of good works) are exponentially distributed with a mean ϵ^{-1}. We consider two schemas: in schema I it is assumed that server might be failure in both cases, i.e. when server in working status and when server is idle; in schema II it is assumed that server might be failure only in case when server is idle. In both schemas, when server fails, it requires an exponentially distributed repair time with mean η^{-1}.

As in [28] it is assumed that services that are interrupted by failures are eventually resumed from the point of interruption or repeated with re-sampling. All inter-arrival, service, operative periods and repair times are independent of each other.

The main objectives of the paper is to develop efficient computational algorithm for calculation of the steady-state probabilities of the cloud system. By solving this problem, we can also easily obtain QoS metrics of the cloud system that are an average values of following quantities: number of servers in working status, number of jobs in system, throughput and response time. Probability of loss of jobs due to buffer overflow is represent interest in given study as well.

4 Generation of Q-Matrix

Firstly we consider Schema I. State of the system is defined by the two-dimensional vector (i, j), where i is the number of operative servers and j is the number of jobs in the system, respectively. Based on the distribution function of the random variables involved in the formation of the model, we determine that the two-dimensional Markov chain (2-D MC) describes the studied system. The set of all possible states of the system, i.e., state space of given 2-D MC is defined as lattice $S = \{0, 1, ..., N\} \times \{0, 1, ..., R\}$ (see Fig. 2).

The transition intensity from the state (i_1, j_1) to the state (i_2, j_2) is denoted as $q((i_1, j_1), (i_2, j_2))$. The combination of these values involves Q-matrix of given 2-D MC and are determined from the following relations (see. Figure 2):

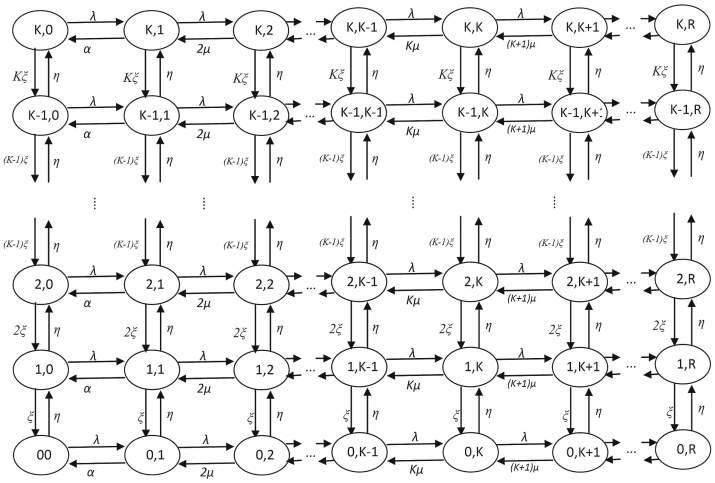

Fig. 2. State diagram of the proposed model

For the case $i_1 = 0$:

$$q((0, j_1), (i_2, j_2)) = \begin{cases} \lambda, & \text{if } (i_2, j_2) = (0, j_1 + 1), \\ \eta, & \text{if } (i_2, j_2) = (1, j_1), \\ j_1\alpha, & \text{if } (i_2, j_2) = (0, j_1 - 1), \\ 0, & \text{in other cases} \end{cases} \tag{1}$$

For the case $i_1 > 0$:

$$q((i_1, j_1), (i_2, j_2)) = \begin{cases} \lambda, & \text{if } (i_2, j_2) = (i_1, j_1 + 1), \\ \eta, & \text{if } (i_2, j_2) = (i_1 + 1, j_1), \\ i_1\xi, & \text{if } (i_2, j_2) = (i_1 - 1, j_1), \\ \min(i_1, j_1)\mu, & \text{if } (i_2, j_2) = (i_1, j_1 - 1), \\ 0, & \text{in other cases} \end{cases} \tag{2}$$

Note that the given finite 2-D MC is an irreducible, so there exists a stationary mode. Let $p(i, j)$ means the stationary probability of state $(i, j) \in S$. These probabilities are determined by solving the system of equilibrium equations (SEE) together the normalizing condition. SEE is compiled on the basis of (1) and (2) and due to their evidence, explicit form of this SEE does not given here.

5 Exact Formulas for QoS Metrics

After the solving SEE, characteristics of the studied system are defined as the marginal distributions of the 2-D MC. Thus, indicated above performance measures of the system are defined as follows:

average number of servers in working status N_w is

$$N_w = \sum_{i=1}^{N} i \sum_{j=0}^{R} p(i,j) \tag{3}$$

average number of jobs in system L_s is

$$L_s = \sum_{j=1}^{R} j \sum_{i=0}^{N} p(i,j) \tag{4}$$

probability of loss of jobs P_b is

$$P_b = \sum_{i=0}^{N} p(i,R) \tag{5}$$

Average throughput TH and average response time RT are calculated as follows:

$$\Gamma_{av} = \mu N_w \tag{6}$$

$$RT = L_s/\Gamma_{av} \tag{7}$$

The dimension of indicated SEE determined by the dimension of the state space S, which is defined as $(N+1)$ $(R+1)$. Unfortunately, it is difficult to find the analytical solution of this system of equations. Matrix-geometric [31] and spectral expansion [29] methods can be employed for the numerical solution of the given problem. However implementation of these methods require additional conditions on the transition rates. First, it's mandatory that transitions rates due to impatience of jobs are independent on number of jobs in queue. However from (1) it's obvious that transitions rates due to impatience of jobs are linear function of number of jobs in queue but not constant one. Secondly, in cloud computing systems numbers of servers and buffer size are in orders of hundreds to thousands. So these methods becomes inefficient due to large dimension of the state space S since their applying causes problems related to ill conditionality of high-dimensional matrices used at different stage of appropriate algorithms.

Therefore, to eliminate computational difficulties, we use the space merging method (SMM) [30] for approximate calculation of the stationary distribution of 2-D MC. This method can be applied here correctly, because, in cloud system servers are reliable in terms of hardware and software configuration. Namely, drop of incoming packet are rarely observed due to high sustainability of the servers. Furthermore, the repair time of the server are measured in hours. However arrival intensity and service intensity of packets much higher than drop intensity or repair intensity of the server [28].

6 Approximate Method

In accordance to the above mentioned comments we concluded that the transitions intensity between the states within the columns in the state diagram is

much higher than the lateral transitions intensity between them (see Fig. 2). Then considering the following splitting of state space S

$$S = \cup_{i=0}^{N} S_i \tag{8}$$

where $S_i = \{(i,j) \in S : j = 0,1,...,R\}, i = 0,1,...,N$. In other words, it is investigated splitting transition diagram by column (see Fig. 2).

Merging function given in state space S is determined based on the splitting (8) as follows:

$$U((i,j)) = <i> \tag{9}$$

where $<i>$ is a merging state, which includes all the states of the class S_i. Let $\Omega = \{<i>: i = 0,1,...,N\}$ According to the space merging method (SMM) algorithms [30], we find that the state probability of the initial model is defined as follows:

$$p(i,j) \approx \rho_i(j)\pi(<i>), \tag{10}$$

where $\rho_i(j)$ denotes the state probability of (i,j) within the splitting model with state space S_i, and $\pi(<i>)$ is the probability of the merging state $<i> \in \Omega$

From splitting scheme (8) it is clear that all the splitting models are one-dimensional birth and death processes (1-D BDP), so that in the class of states S_i the first component is constant. Therefore, in the study of the splitting model with state space S_i microstate $(i,j) \in S$ of given model can be represent by scalar $j, j = 0,1,...,R$. From (2) and (3) we get that these parameters for the splitting model with state space S_i are defined as follows (see Fig. 2):

For the case $i = 0$:

$$\rho_0(j) = \frac{\frac{\nu_1^j}{j!}}{\sum_{i=0}^{R} \frac{\nu_1^i}{i!}}, j = 0,1,...,R, \text{where } \nu_1 = \lambda/\alpha; . \tag{11}$$

For the case $i = 1,...,N$

$$\rho_i(j) = \begin{cases} \frac{\nu_2^j}{j!}\rho_i(0), & \text{if } 0 \le j \le i, \\ \frac{\nu_2^j}{j!i^{j-i}}\rho_i(0), & \text{if } i+1 \le j \le R, \end{cases} \tag{12}$$

where $\nu_2 = \lambda/\mu$ and $\rho_i(0)$ is calculated from normalizing condition, i.e. $\sum_{j=0}^{R} \rho_i(j) = 1$.

The transition intensity from the merging state $<i_1>$ to other merging state i_2 is denoted $q(<i_1>,<i_2>), <i_1>, <i_2> \in \Omega$ Then after certain algebras on the bases of (1), (2), (11) and (12) we obtain:

$$q(<i_1>,<i_2>) = \begin{cases} \eta, & \text{if } i_2 = i_1 + 1, \\ i_1\xi, & \text{if } i_2 = i_1 + 1 \\ 0, & \text{in other cases.} \end{cases} \tag{13}$$

From (13) we obtain that the merging state probabilities $\pi(<j>), <j> \in \Omega$ are calculated as the state probabilities of classical Erlang's model $M/M/N/N$

with load $\sigma = \eta/\xi$ erl, i.e.

$$\pi_1(< j >) = \frac{\frac{\sigma^j}{j!}}{\sum_{i=0}^{N} \frac{\sigma^i}{i!}}, j = 0, 1, ..., N \tag{14}$$

Therefore, taking into account the relations (10)-(14) we obtain steady-state probabilities $p(i, j), (i, j) \in S$.

After certain algebras we obtain the following simple approximate formulas for calculating the desired QoS metrics of the cloud system:

$$N_w \approx \sigma(1 - E_B(\sigma, N)) \tag{15}$$

where $E_B = \frac{\frac{\sigma^N}{N!}}{\sum_{i=0}^{N} \frac{\sigma^i}{i!}}$ is Erlang's B-formula

$$L_s = \sum_{j=1}^{R} j \sum_{i=0}^{N} \rho_i(j)\pi_1(<i>) \tag{16}$$

$$P_b = \sum_{i=0}^{N} \rho_i(R)\pi_1(<i>) \tag{17}$$

From (15)–(17) by using (6) and (7) performance measures TH and RT might be calculated.

Note 1. Formula (15) indicated that an average number of servers in working status does not dependent on load parameters of incoming jobs λ and μ and it is determined by failure and repair rates ξ and η. This result was expected one since in Schema I the events failure and repair of servers does not depend on number of jobs in cloud system. In other words, by using approximate approach we have find exact result for one of performance measure. This is undirected proof of high accuracy of proposed approach.

Now consider Schema II. Since the state diagram of the Schema II is very similar to the diagram of Schema I, we didn't included the diagram of Schema II. In this schema elements of Q-matrix for the case $i_1 = 0$ are calculated by the formula (1) as well. However in this schema elements of Q-matrix for the case are calculated as follows:

$$q((i_1, j_1), (i_2, j_2)) = \begin{cases} \lambda, & \text{if } (i_2, j_2) = (i_1, j_1 + 1), \\ \eta, & \text{if } (i_2, j_2) = (i_1 + 1, j_1), \\ (i_1 - j_1)^+\xi, & \text{if } (i_2, j_2) = (i_1 - 1, j_1), \\ \min(i_1, j_1)\mu, & \text{if } (i_2, j_2) = (i_1, j_1 - 1), \\ 0, & \text{in other cases.} \end{cases} \tag{18}$$

From (1) and (18) we can obtain SEE for the Schema II. In order to solve computational difficulties of the steady-state probabilities of this Schema, we can apply proposed above approximate method. No repeating above procedures

note that in this case steady-state probabilities within splitting models are calculated by (11) and (12) also. However in this case merging state probabilities $\pi_2(<j>), <j> \in \Omega$ are calculated as follows:

$$\pi_2(<j>) = \frac{\sigma^j}{\prod_{i=1}^{j} \theta(i)} \pi_2(<0>), j = 1, ..., N \tag{19}$$

where $\pi_2(<0>)$ is calculated from normalizing condition, i.e.

$\pi_2(<0>) = \left(1 + \sum_{j=1}^{N} \frac{\sigma^j}{\prod_{i=1}^{j} \theta(i)}\right)^{-1}$ and $\theta(i) = \sum_{j=1}^{i} j\rho_i(i-j)$.

QoS metrics L_s and P_b are calculated similar to (16) and (17) where $\pi_1(<j>)$ are substituted by $\pi_2(<j>)$ but N_w is calculated as follows:

$$N_w \approx \sum_{i=1}^{N} i\pi_2(<i>) \tag{20}$$

As we seen from formula (20) in this schema an average number of servers in working status depend on load parameters of incoming jobs λ and μ as well as on failure and repair rates ξ and η.

7 Numerical Results

As numerical results we have calculated exact and approximate solution and compared them. From (1) and (2) we get exact values of the system at the given values of the parameters $R = 100$, $N = 50$, $\lambda = 50, \alpha = 0.005, \mu = 1, \xi = 0.008, \eta = 0.0001$. Exact values were driven from Matlab 9.0 running on the personal computer with Intel i7 processor and 16 GB of RAM. Since size of matrix is equal to $(N+1)(R+1)$, it is too difficult to calculate exact values by using SEE at higher value of N and R.

In order to compare an exact and approximate values, first of all, we took comparison of $p(i, j)$ (steady-state probabilities) as an absolute and cosine values (see Table 1). As we mentioned above, for lower values it is possible to calculate exact values of the steady-state probabilities, but for big number it is impossible. Our approximate methods calculate these probabilities at higher values of N and R. In real cloud computing system always involves many server. This means, in real practice we always have to take into consideration a large amount of servers.

Accuracy of the approximate method to calculation of steady-state probabilities is estimated by two norms: (1) Absolute value of maximum of differences between values of steady-state probabilities (Norm 1); (2) cosine similarity (Norm 2) (see Table 1).

High arrival intensity of requests shouldn't dramatically affect the general performance of the cloud system with N servers. For a certain number of server, the higher arrival intensity means that request will wait in the queue, but after all they will be served.

The same behavior can be seen in the nature of the Schema II. However, in comparison with the Schema I, we get more optimal QoS metric in the same

Table 1. Values of various norms for Schema I

R	50	50	50	50	100	100	100	100	200	200	200	200	500	500	500	500
N	10	20	30	40	10	20	30	40	20	40	60	80	20	40	60	80
Norm 1	0.00545	0.00645	0.01246	0.01046	0.09345	0.09011	0.08351	0.09022	0.09055	0.08625	0.08254	0.08952	0.07854	0.0825	0.08625	0.07905
Norm 2	0.9108	0.9276	0.9404	0.9003	0.9035	0.9034	9.062	0.9118	0.9331	0.9099	0.9078	0.9045	0.9339	0.9782	0.9325	0.9225

initial values. Although, there isn't a big difference in QoS metrics between Schema I and II, this differences make huge values in real practice in terms of cost efficiency.

Tables 2 and 3 gives exact and approximate values of QoS metrics for Schema I and Schema II, respectively. Because of multiplication operation in the QoS metrics formulas we have a little bit distinguish, that can be assumed as the calculation error. Probability of blocking for almost the same. The bigger number of server in the cloud system, the lower the loss of the requests. It doesn't mean that we should have a number of server in the cloud system. The point here is that we should utilize the servers as much as optimal in order to reduce the loss of request. The same scenario can be said for the Schema II.

Table 2. Exact and approximate value of QoS metrics for Schema I, $R = 100$, *– Exact Values, **– Approximate Values

N	N_w^*	N_w^{**}	L_s^*	L_s^{**}	Γ_{av}^*	Γ_{av}^{**}	RT^*	RT^{**}	PB^*	PB^{**}
6	5.513083	5.921185	8.523196	9.15412	5.513083	5.921185	1.439441	1.545994	0.00015	1.61E−04
11	10.07285	10.84566	10.36	11.15485	10.07285	10.84566	0.95522	1.028508	3.09E−05	3.32E−05
16	14.59946	15.75917	14.41439	15.55939	14.59946	15.75917	0.914667	0.987323	2.08E−05	2.24E−05
21	19.09073	20.65924	18.61579	20.14527	19.09073	20.65924	0.901088	0.975122	7.21E−06	7.79E−06
26	24.32102	25.54262	23.52669	24.70838	24.32102	25.54262	0.921076	0.96734	4.95E−07	5.36E−07
31	30.87354	30.40499	29.6279	29.17826	30.87354	30.40499	0.974442	0.959653	1.13E−07	1.18E−07
36	35.80708	35.24053	34.02942	33.491	35.80708	35.24053	0.965633	0.950354	6.82E−08	6.71E−08
41	40.71159	40.04118	38.18912	37.56024	40.71159	40.04118	0.953746	0.93804	5.45E−08	5.36E−08

Table 3. Exact and approximate value of QoS metrics for Schema II, $R = 100$,*– Exact Values, **– approximate values

N	N_w^*	N_w^{**}	L_s^*	L_s^{**}	Γ_{av}^*	Γ_{av}^{**}	RT^*	RT^{**}	PB^*	PB^{**}
6	5.576369	5.989155	8.447154	9.072448	5.576369	5.989155	1.41041	1.514813	0.000167335	1.80E−04
11	10.12505	10.90187	10.40127	11.19928	10.12505	10.90187	0.95408	1.02728	6.50718E−05	7.01E−05
16	14.64189	15.80497	14.45358	15.6017	14.64189	15.80497	0.9145	0.987138	2.92429E−05	3.16E−05
21	19.13031	20.70207	18.65292	20.18546	19.13031	20.70207	0.90102	0.975045	1.08088E−06	1.17E−06
26	24.3647	25.58849	23.56736	24.7511	24.3647	25.58849	0.92101	0.967275	1.62671E−06	1.71E−06
31	30.93018	30.46078	29.6796	29.22917	30.93018	30.46078	0.97435	0.959568	7.03302E−07	6.93E−07
36	35.88418	35.31641	34.09742	33.55792	35.88418	35.31641	0.96548	0.950208	6.40072E−08	6.30E−08
41	40.82667	40.15436	38.28501	37.65455	40.82667	40.15436	0.95345	0.937745	1.12406E−08	1.11E−08

High arrival intensity of requests shouldn't dramatically affect the general performance of the cloud system with N servers. For a certain number of server,

the higher arrival intensity means that request will wait in the queue, but after all they will be served. Total response time is acceptable. The same behavior can be seen in the nature of the Schema II. However, in comparison with the Schema I, we get more optimal QoS metric in the same initial values. Although, there isn't a big difference in QoS metrics between Schema I and II, this differences make huge values in real practice in terms of cost efficiency. For example, at the given initial values we have around 10 server difference in the values of Nav. This 10 servers implies extra budget for the cloud system.

8 Conclusion and Future Works

As we mentioned above modern web servers are extremely distinguished from small server to the giant computing system depending on services they're performing and number of users, because of network capabilities some arrival packet may be dropped because of no idle place in the processing queue, even though the server is under fairly light load. Here we considered queuing management in cloud computing centers with large numbers of web servers. Our contribution in this kind research is feasibility of implementation of state merging algorithm because of unit difference in the server repair time and interarrival times of jobs. Namely, server repair time is calculated in hours, where interarrival times of jobs is with seconds. Our result is applicable in the real cloud system in order to calculate the QoS metrics depending of application area.

References

1. Briscoe, G., Marinos, A.: Digital ecosystems in the clouds: towards community cloud computing. In: Proceedings of 3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST 2009), pp. 103–108. IEEE (2009)
2. Marinos, A., Briscoe, G.: Community cloud computing. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 472–484. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10665-1_43
3. Cardellini, V., Casalicchio, E., Colajanni, M.: A performance study of distributed architectures for the quality of web services. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34), Washington, DC, USA, vol. 9, p. 9019. IEEE Computer Society (2001)
4. Athula, G., Murugesan, G.: Guest editors' introduction Web engineering: an introduction. IEEE MultiMedia **8**(1), 14–18 (2001)
5. Mei, R.D., Hariharan, R., Reeser, P.K.: Web server performance modeling. Telecommun. Syst. **16**(3–4), 361–378 (2001)
6. Asho, E., Onuttom, N., Walter, W.: Experimental queueing analysis with long-range dependent packet traffic. IEEE/ACM Trans. Netw. **4**(2), 209–223 (1996)
7. Cao, J., Andersson, M., Nyberg, C., Kihl, M.: Web server performance modeling using an M/G/l/K*PS queue. In: 10th International Conference on ICT 2003, vol. 2, pp. 1501–1506, February/March 2003
8. Xiong, K., Perros, H.: Service performance and analysis in cloud computing. In: Proceedings of IEEE World Conference Services, pp. 693–700 (2009)

9. Slothouber, L.: A model of web server performance. In: Proceedings of the Fifth International World Wide Web Conference (1996)
10. Yang, B., Tan, F., Dai, Y.-S., Guo, S.: Performance evaluation of cloud service considering fault recovery. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 571–576. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10665-1_54
11. Ma, N., Mark, J.: Approximation of the mean queue length of an M/G/c queueing system. Oper. Res. **43**, 158–165 (1998)
12. Karlapudi, H., Martin, J.: Web application performance prediction. In: Proceedings of the IASTED International Conference on Communication and Computer Networks, pp. 281–286 (2009)
13. Mei, R.D., Meeuwissen, H.B.: Modelling end-to-end Quality-of-Service for transaction-based services in multidomain environment. In: Proceedings of the 19th International Teletraffic Congress (ITC 19), pp. 1109–1121 (2005)
14. Meisner, D., Gold, B.T., Wenisch, T.F.: PowerNap: eliminating server idle power. SIGPLAN Not. **44**, 205–216 (2009)
15. Meisner, D., Sadler, C.M., Barroso, L.A., Weber, W.-D., Wenisch, T.F.: Power management of online data-intensive services. In: Iyer, R., Yang, Q., González, A. (eds.) ISCA, pp. 319–330. ACM (2011)
16. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. **41**(1), 23–50 (2011)
17. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: the montage example. In: SC-International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–12 (2008)
18. Bruneo, D.: A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. IEEE Trans. Parallel Distrib. Syst. **25**(3), 560–569 (2014)
19. Cao, J., Hwang, K., Li, K., Zomaya, A.Y.: Optimal multiserver configuration for profit maximization in cloud computing. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1087–1096 (2013)
20. Chiang, Y.J., Ouyang, Y.C., Hsu, C.H.: An efficient green control algorithm in cloud computing for cost optimization. IEEE Trans. Cloud Comput. **3**(2), 145–155 (2015)
21. Ghosh, R., Longo, F., Naik, V.K., Trivedi, K.S.: Modeling and performance analysis of large scale IaaS clouds. Future Gen. Comput. Syst. **29**(5), 1216–1234 (2015)
22. Jin, Y., Wen, Y., Zhang, W.: Content routing and lookup schemes using global bloom filter for content-delivery-as-a-service. IEEE Syst. J. **8**(1), 268–278 (2014)
23. Khazaei, H., Misic, J., Misic, V.: Performance analysis of cloud computing centers using M/G/m/m+r queuing systems. IEEE Trans. Parallel Distrib. Syst. **23**(5), 936–943 (2012)
24. Mei, J., Li, K., Ouyang, A., Li, K.: A profit maximization scheme with guaranteed quality of service in cloud computing. IEEE Trans. Comput. **64**(11), 3064–3078 (2015)
25. Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Abella, F., Rius, J.: A queuing theory model for cloud computing. J. Supercomput. **69**(1), 492–507 (2014)
26. Yang, B., Tan, F., Dai, Y.S.: Performance evaluation of cloud service considering fault recovery. J. Supercomput. **65**(1), 426–444 (2013)
27. János, S., Bérczes, T.: Tool supported analysis of queueing systems with Future Internet applications. In: Pre-Proceedings of 9th International Conference on Applied Mathematics, Baia Mare, Romania, pp. 114–117 (2013)

28. Enver, E.J.: Performability analysis of cloud computing centers with large number of servers. J. Supercomput. **73**(5), 2130–2156 (2017)
29. Chakka, R.: Spectral expansion solution for some finite capacity queues. Ann. Oper. Res. **79**, 27–44 (1998)
30. Ponomarenko, L., Kim, C.S., Melikov, A.Z.: Performance Analysis and Optimization of Multi-Traffic on Communication Networks. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15458-4
31. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach, p. 332. John Hopkins University Press, Baltimore (1981)