Performance Analysis of Cloud Centers under Burst Arrivals and Total Rejection Policy

Hamzeh Khazaei
University of Manitoba, Winnipeg, Canada

Jelena Mišić
Ryerson University, Toronto, Canada

Vojislav B. Mišić
Ryerson University, Toronto, Canada

***Abstract*—Quality of service, QoS, has a great impact on wider adoption of cloud computing. Maintaining the QoS at an acceptable level for cloud users requires an accurate and well adapted performance analysis approach. In this paper, we describe a new analytical model for performance evaluation of cloud server farms under burst arrivals and solve it to obtain important performance indicators such as mean request response time, blocking probability, probability of immediate service and probability distribution of number of tasks in the system. This model allows cloud operators to tune the parameters such as the number of servers and/or burst size, on one side, and the values of blocking probability and probability that a task request will obtain immediate service, on the other.**

I. Introduction and Related Work

Cloud Computing is a computing paradigm in which different computing resources such as infrastructure, platforms and software applications are made accessible over the internet to remote user as services [1].

Due to dynamic nature of cloud environments, diversity of user's requests and time dependency of load, providing expected quality of service while avoiding over-provisioning is not a simple task [2]. To ensure that the QoS perceived by end clients is acceptable, the providers must exploit techniques and mechanisms that guarantee a minimum level of QoS. Although QoS has multiple aspects such as response time, throughput, availability, reliability, and security, the primary aspect of QoS considered in this work is related to response time [3].

In this paper we describe an analytical model for evaluating the performance of cloud server farms under burst arrival and verify its accuracy with numerical calculations and simulations. We assume that a user may submit a burst of tasks all together; we refer the whole burst as super-task in this context; any super-task goes through a *facility node* and then leaves the center. A facility node may contain different computing resources such as web servers, database servers, directory servers and others. We adopt the *total rejection policy* for both admission and servicing; if there is enough room for whole super-task then it will be accepted otherwise the super-task will be lost. The same scenario is also adopted for servicing; all tasks within super-task will get into service at the same time; in other words, super-task will get into service if there is enough idle server for whole tasks within the super-task.

We model the cloud environment as an $M^{[x]}/G/m/m+r$ queuing system which indicates that inter-arrival time of super-tasks is exponentially distributed, the service time of each task in super-tasks is generally distributed, the number of facility nodes is m and the capacity of system is equal to $m+r$; moreover the probability distribution of super-task size, number of tasks within super-task, is also generally distributed.

These two characteristics, generally distributed service time and large number of nodes, have not been adequately addressed in previous research [4]. To the best of our knowledge there is no research on performance analysis of cloud centers under burst arrival. In this paper we incorporate all the inevitable characteristics of a typical cloud center; burst arrival, generally distributed service time, large number of servers and finite capacity have been taken into account for performance modelling.

Analysis in the case where either inter-arrival or service times (or both) are not exponential is complex; incorporating burst arrival makes the analysing even more complicated. It is known that even for the simplest case, the $M^{[x]}/M/m$ queue, no closed-form results exist. As a result the probability distributions of response time and queue length in $M^{[x]}/G/m/m+r$ cannot be obtained in closed-form, which necessitated the search for a suitable approximation.

An upper bound for the mean queue length and lower bounds for the delay probabilities (that of an arrival burst and that of an arbitrary task in the arrival burst) was described in [5]. An approximate formula is also developed for the general burst-arrival queue $GI^{x}/G/m$. In spite of the simplicity and acceptable performance, the approach is accurate enough for small burst sizes, up to 3, as well as small number of servers (less than 10). In [6], the authors proposed an approximation method for the computation of the steady-state distribution of the number of tasks in queue as well as the moments of the waiting time distribution. They examined both hypo-exponential and hyper-exponential distribution family for service time, which is necessary for modelling a dynamic system such as cloud farms; however they just performed numerical results for a system with up to seven server and there is no result or indication about the efficiency of the method in case of larger number of servers or a system with finite capacity. Another approximate formula was proposed in [7]. The authors presented an approximate formula for the steady-state average number of tasks in the $M^{[x]}/G/m$ queuing system. The derivation of the formula is based on a heuristic argument whereby a reformulation of the number of tasks in $M^{[x]}/G/1$ is extended to the multi-server queue. From a computational viewpoint, the approach is simple to apply, though, the relative percentage error incurred seem to be unavoidable when the number of server is large, the mean burst size is small or the *coefficient of variation*, CoV, defined as the ratio of standard deviation to mean value, of service time is bigger than one.

A diffusion approximation for an $M/G/m$ queue with burst arrival was developed in [8]. The authors derived an approximate formula for the steady-state distribution of the number of tasks in the system, delay probability and mean queue length. Diffusion approach is totally inaccurate when the burst size is relatively large (more than 3).

II. Analytical Model

We adopt $M^{[x]}/G/m/m+r$ queuing system as the abstract model for the performance analysing. Like our previous work [4], we use Embedded Markov chain technique for analysing the resulted system. We look into system at moment of super-task arrival and find the steady-state distribution of number of tasks in system at arrivals. Due to the absence of PASTA [9], we then find the steady-state distribution of number of tasks in system using Semi-Markov process at arbitrary time.

Let g_k be the probability that the super-task size is equal to k, $k = \{1, 2, ..., MBS\}$ in which MBS is the *Maximum Burst Size* that we assume for our system. Let $\overline{g}$ and G be the mean value and *Cumulative Distribution Function*, CDF, of X_g respectively.

$$\begin{aligned} g_k &= Prob[X_g = k] \quad k = 1, 2, ..., MBS \\ G(k) &= Prob[X_g \leq k] = \sum_{i=1}^{k} g_i \\ \overline{g} &= E[X_g] \end{aligned} \tag{1}$$

and here we set MBS as $MBS = 2\overline{g} + 1$. Super-task request arrivals follow a Poisson process so super-task request inter-arrival time A is exponentially distributed with rate of $\frac{1}{\lambda}$. We denote its CDF as $A(x) = Prob[A < x]$ and its *Probability Density Function* (pdf) as $a(x) = \lambda e^{-\lambda x}$. *Laplace Stieltjes Transform* (LST) of inter-arrival time is

$$A^*(s) = \int_0^\infty e^{-sx} a(x) dx = \frac{\lambda}{\lambda + s}$$

Each task within super-task has a service times which identically and independently distributed according to a general distribution B, with a mean service time equal to $\overline{b} = \frac{1}{\mu}$. The CDF of the service time is $B(x) = Prob\left[B < x\right]$, and its pdf is $b(x)$. The LST of service time is

$$B^*(s) = \int_0^\infty e^{-sx} b(x) dx$$

Residual task service time is the time interval from an arbitrary point (an arrival point in a Poisson process) during a service time to the end of the service time; we denote it as B_+. *Elapsed task service time* is the time interval from the beginning of a service time to an arbitrary point of the service time; we denote it as B_-. It can be shown that probability distribution of residual and elapsed task service times has the same probability distribution and LST of them can be calculated as [9]

$$B_+^*(s) = B_-^*(s) = \frac{1 - B^*(s)}{s\bar{b}} \tag{2}$$

The *traffic intensity* may be defined as $\rho \triangleq \frac{\lambda \overline{g}}{m\mu}$. which for practical reason we assume that $\rho < 1$. We consider total rejection policies for admission and servicing [9]; that is at the moment of super-task arrival if the system doesn't have enough room for whole super-task, the super-task would be lost; and the service time of whole tasks in a super-task start at the same time on different servers; in the other words, if number of idle servers is less than super-task size then those idle servers remain unused until other servers become free and then the super-task can be fitted in servers. Here the waiting time for the first, last and any arbitrary tasks in a super-task is identical because all of them get into service at the same time.

A. The Markov chain

We are looking at the system at the moments of super-task arrivals – these points are selected as Markov points. The Markov chain of the system is shown in Fig. 1.

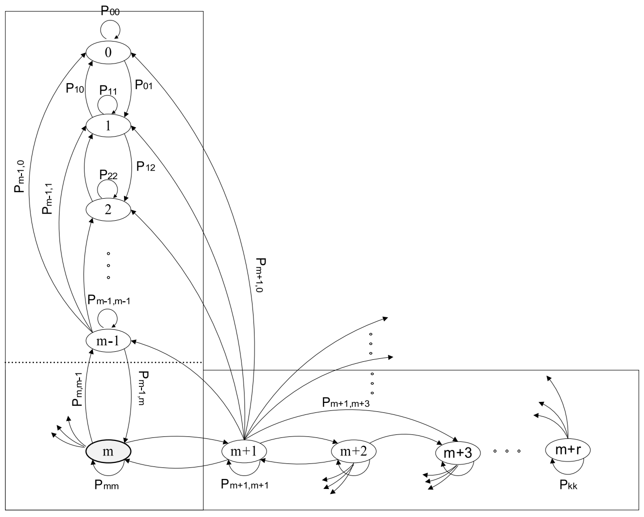

Fig. 1. Markov Chain of $M^{[x]}/G/m/m+r$ queuing system

Let A_n and A_{n+1} indicate the moment of n^{th} and $(n+1)^{th}$ arrivals to the system, respectively, while q_n and q_{n+1} indicate the number of tasks found in the system immediately before these arrivals; this is schematically shown in Fig.2. If k is the size of super-task and v_{n+1} indicates the number of tasks which depart from the system between A_n and A_{n+1}, then we have: $q_{n+1} = q_n - v_{n+1} + k$

We need to calculate the transition probabilities associated with the Markov chain, defined as

$$P(i, j, k) \triangleq Prob\left[q_{n+1} = j | q_n = i \text{ and } X_g = k\right] \tag{3}$$

i.e., the probability that $i + k - j$ customers are served during the interval between two successive task request arrivals.

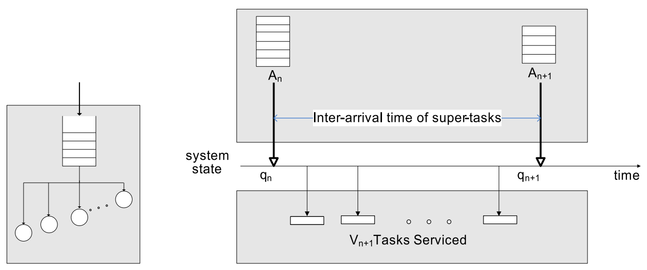

Fig. 2. Markov points for $M^{[x]}/G/m/m+r$ queuing system

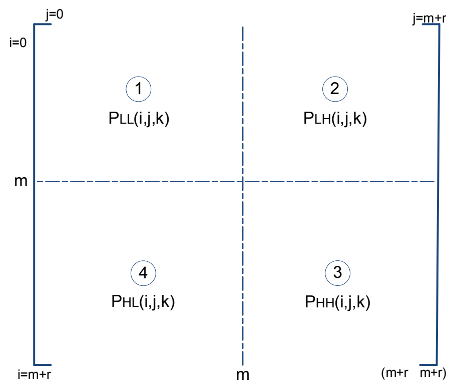

Fig. 3. One-step transition probability matrix: Range of validity for p_{ij} equations.

Obviously for $j > i+k \Rightarrow P(i,j,k) = 0$; since there are at most $i+k$ tasks present between the arrival of A_n and A_{n+1}. The Markov state-transition-probability diagram is shown in Fig. 1, where states are numbered according to the number of tasks currently in the system (i.e., those in service and those awaiting service). For clarity, some transitions are not fully drawn. We have also highlighted the state m because the transition probabilities are different for states on the above and right hand side of this state.

B. Departure Probabilities

To find the elements of the transition probability matrix, we need to count the number of tasks departing from the system in between two successive arrivals. Each server has zero or more departures during the time between two successive super-task arrivals (the inter-arrival time). Let us focus on an arbitrary server; for a task to finish and depart from the system during the inter-arrival time, its remaining duration (residual service time defined in (2)) must be shorter than the task inter-arrival time. This probability will be denoted as

$$\begin{aligned} P_x &= Prob\left[A > B_+\right] = \int_{x=0}^{\infty} P\{A > B_+ | B_+ = x\} dB_+(x) \\ &= \int_{x=0}^{\infty} \left(\int_{y=x}^{\infty} \lambda e^{-\lambda y} dy \right) dB_+(x) \\ &= \int_{0}^{\infty} e^{-\lambda x} dB_+(x) = B_+^*(\lambda) \end{aligned} \tag{4}$$

In the case when arriving super-task can be accommodated immediately by idle servers (and therefore queue length is zero) we have to evaluate the probability that such task will depart before next super-task arrival. We will denote this probability as P_y and calculate it as:

$$\begin{aligned} P_y &= Prob\left[A > B\right] = \int_{x=0}^{\infty} P\{A > B | B = x\} dB(x) \\ &= \int_{x=0}^{\infty} \left(\int_{y=x}^{\infty} \lambda e^{-\lambda y} dy \right) dB(x) \\ &= \int_{0}^{\infty} e^{-\lambda x} dB(x) = B^*(\lambda) \end{aligned} \tag{5}$$

However, if queue is non-empty upon super-task arrival, the following may happen. If between two successive super-task arrivals a completed task departs from a server, that server along with other idle servers might take a new super-task from the non-empty queue. That task may be completed as well before the next super-task arrival and if the queue is still non-empty new task may be executed, and so on until either queue gets empty or new super-task arrives. Therefore probability of $k > 0$ job departures from a single server, given that there are enough jobs in the queue can be derived from expressions (4) and (5) as:

$$P_{z,k} = B_+^*(\lambda)(B^*(\lambda))^{k-1}, \quad 0 < k \leq m+r \tag{6}$$

Note that $P_{z,1} = P_x$.

Using these values we are able to compute the transition probabilities matrix.

C. Transition Matrix

Based on our Markov chain, and servicing policy we may identify four different regions of operation for which different conditions hold; these regions are schematically shown in Fig. 3 as one-step transition probability matrix. The numbers on rows and columns correspond to the number of tasks in the system immediately before a super-task arrival (i) and immediately upon the next super-task arrival (j), respectively. We also have k in transition probability which indicate the size of super-task.

1) Transition Probabilities:

- Regarding the region labelled 1, the transitions originate and terminate on the left hand side of state m indicating that queue is empty. for $i, j \leq m$ we have:

$$P_{LL}(i,j,k) = \begin{cases} \displaystyle\sum_{z=0}^{min(i,i+k-j)} \binom{i}{z} P_x^z (1-P_x)^{i-z} \cdot \\ \binom{k}{i+k-j-z} P_y^{(i+k-j-z)} (1-Py)^{(z-i+j)}, & \text{if } i+k \leq m \\ \displaystyle\sum_{z=i+k-m}^{min(i,i+k-j)} \binom{i}{z} P_x^z (1-P_x)^{i-z} \cdot \\ \binom{k}{i+k-j-z} P_{z,2}^{(i+k-j-z)} (1-P_{z,2})^{(z-i+j)}, & \text{if } i+k > m \end{cases} \tag{7}$$

- In Region 2, the transitions originate from the left hand side of state m and then fall into the right hand side of state m. In other words, at starting state the queue is empty while at the ending state the queue is not empty. In this case the arriving super-task cannot be accommodated in the idle servers so it will be queued.

$$P_{LH}(i,j,k) = \binom{i}{i+k-j} P_x^{(i+k-j)}(1-P_x)^{j-k} \quad \text{for } i < m, j > m \tag{8}$$

- Region 3 corresponds to the case where the queue is not empty throughout the inter-arrival time, i.e., $i, j > m$. Let us denote the number of jobs which depart from the system between these two Markov points as $w(k) = i + k - j$. We also define the probability $P_a(t,k)$ that is the probability of having k active server out of t server.(note that in total rejection policy we may have some idle server even thought the queue is not empty).

$$P_a(t,k) = \begin{cases} G(MBS) - G(t-k), & k < t \\ 1 - \displaystyle\sum_{z=max(0,t-MBS+1)}^{t-1} P_a(t,z), & k = t \end{cases} \tag{9}$$

In this case no transition starts or ends at a state below m in Fig. 1, and the state transition probabilities can be computed for $i, j > m$ as:

$$\begin{aligned} P_{HH}(i,j,k) = & \sum_{\psi=(m-MBS+1)}^{m} P_a(m,\psi) \cdot \\ & \sum_{s_1=\min(w(k),1)}^{\min(w(k),\psi)} \binom{\psi}{s_1} P_x^{s_1}(1-P_x)^{(\psi-s_1)} \cdot \\ & \sum_{s_2=\min(w(k)-s_1,1)}^{\min(\alpha,w(k)-s_1)} \binom{\alpha}{s_2} P_{z,2}^{s_2}(1-P_{z,2})^{(\alpha-s_2)} \cdot \\ & \binom{s_2}{\beta} P_{z,3}^{\beta}(1-P_{z,3})^{(s_2-\beta)} \\ & \alpha = m - \psi + s_1 \;\&\; \beta = w(k) - s_1 - s_2 \end{aligned} \tag{10}$$

Note that under moderate load it is not likely to have more than a couple of task departures from a single server.

- Finally, region 4, in which $i > m$ and $j \leq m$, describes the situation where the first arrival (A_n) finds non-empty queue which it joins while at the time of the next arrival (A_{n+1}) there are j tasks in the system, all of which are in service and the system has at least one idle server. The transition probabilities for this region, $i > m, j \leq m$ are

$$\begin{aligned} P_{HL}(i,j,k) = & \sum_{\psi=(m-MBS+1)}^{m} P_a(m,\psi) \cdot \\ & \sum_{s_1=\min(w(k),\psi-j)}^{\min(w(k),\psi)} \binom{\psi}{s_1} P_x^{s_1}(1-P_x)^{(\psi-s_1)} \cdot \\ & \sum_{s_2=\min(w(k)-s_1,\psi-j)}^{\min(\alpha,w(k)-s_1)} \binom{\alpha}{s_2} P_{z,2}^{s_2}(1-P_{z,2})^{(\alpha-s_2)} \cdot \\ & \binom{s_2}{\beta} P_{z,3}^{\beta}(1-P_{z,3})^{(s_2-\beta)} \\ & \alpha = m - \psi + s_1 \text{ and } \beta = w(k) - s_1 - s_2 \end{aligned} \tag{11}$$

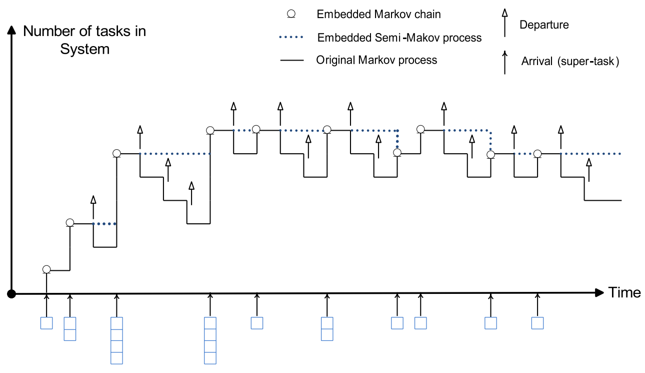

Fig. 4. Original Markov process, semi-Markov process and embedded Markov chain.

III. Equilibrium Balance Equations

After finding matrix $\mathbf{P}$ we can establish the balance equations. The balance equations are:

$$\pi_i = \sum_{j=0}^{m+r} \pi_j p_{ji}, \quad 0 \leq i \leq m+r \tag{12}$$

and augmented by the normalization equation $\sum_{i=0}^{m+r} \pi_i = 1$. So far we have $m + r + 2$ equations which includes $m + r + 1$ linearly independent equations from (12) and one normalization equation; however we have $m + r + 1$ variables $[\pi_0, \pi_1, \pi_2, \ldots, \pi_{m+r}]$; so in order to obtain the unique equilibrium solution we need to remove one of the equations; the wise choice would be the last equation in (12) due to minimum information this equation holds about the system in comparison with the others. Here, the steady state balance equations can't be solved in closed form, hence we must resort to a numerical solution.

A. Distribution of Number of Tasks in the System

Once we obtain the steady state probabilities we are able to establish the probability generating functions (PGFs) for the number of tasks in the system at the time of a super-task arrival: $\Pi(z) = \sum_{k=0}^{m+r} \pi_z z^k$. Due to burst arrival, the PASTA property doesn't hold; thus, the distribution of number of tasks in system at arrival times is not the same with distribution of number of tasks in system at any arbitrary time.

B. Steady-state Distribution at any Arbitrary Time

In order to obtain steady-state distribution of number of tasks at arbitrary time, we employ semi-Markov process [9]. A semi-Markov process imitates the original Markov process but it will be updated just at the Markov points; in other words, the value of semi-Markov process remains constant from one Markov point to the next. Fig. 4 plots original Markov process, semi-Markov process and imbedded Markov chain of our system. Let $H_k(x)$ be the CDF of the residence time that the semi-Markov process is in state k:

$$H_k(x) \triangleq Prob[t_{n+1} - t_n \leq x \mid q_n = k] = 1 - e^{\lambda x} \quad \text{for } k = 0, 1, 2, ..., m+r \tag{13}$$

which in our system does not depend on n. The mean residence time in state k is

$$\overline{h}_k = \int_0^\infty [1 - H_k(x)]dx = \frac{1}{\lambda} \quad k = 0, 1, 2, ..., m + r \quad (14)$$

then the steady-state distribution in the semi-Markov process is given by [9]

$$p_k^{sm} = \frac{\pi_k \overline{h}_k}{\sum_{j=0}^{m+r} \pi_j \overline{h}_j} = \frac{\pi_k}{\lambda \sum_{j=0}^{m+r} \pi_j 1/\lambda} = \pi_k \quad (15)$$

where $\{\pi_k;\ k = 0, 1, \ldots, m + r\}$ is the distribution of the embedded Markov chain. so the steady-state probability of semi-Markov process is identical with the embedded Markov chain. Now we define the the CDF for the time back to the most recent Markov point looking form an arbitrary time by

$$H_k^-(y) = \frac{1}{\overline{h}_k} \int_0^y [1 - H_k(x)]dx \quad k = 0, 1, 2, ..., m + r \quad (16)$$

The arbitrary-time distribution is given by

$$p_i = \sum_{j=i}^{m+r} p_j^{sm} \cdot \int_0^\infty \text{Prob[moving from } j \text{ to } i \text{ during } y\ ]\, dH_j^-(y) = \sum_{j=i}^{m+r} \pi_j P(j, i, 0) \quad (17)$$

The PGF of the number of tasks in system is given by

$$P(z) = \sum_{i=0}^{m+r} p_i z^i \quad (18)$$

Mean number of tasks in the system, then, obtained as:

$$\overline{p} = P^{'}(1) \quad (19)$$

C. *Blocking Probability and Mean Response Time*

Since arrivals are independent of buffer state and the distribution of number of tasks in the system was obtained, we are able to directly calculate the blocking probability of a super-task in the system with buffer size of r:

$$P_b = \sum_{k=0}^{MBS-1} \left[\sum_{i=0}^{MBS} p_{m+r-i-k}(1 - G(i)) \right] \cdot P_a(m, m-k) \quad (20)$$

The appropriate buffer size, r_ϵ, in order to have the blocking probability below the certain value, ϵ, is:

$$r_\epsilon = min\{r \geq 0 \mid P_b \leq \epsilon\} \quad (21)$$

The effective arrival rate to the system can be calculated as $\lambda_e = (1 - P_b)\lambda\overline{g}$. By Little's law, the mean response time is obtained as:

$$\overline{t} = \frac{\overline{p}}{\lambda_e} \quad (22)$$

D. *Probability of Immediate Service*

Here we are interested in the probability with that super-tasks will get into service immediately upon arrival, without any queueing. For such super-tasks, the response time would be equal to the service time:

$$P_{nq} = \sum_{k=0}^{MBS-1} P_a(m, m-k) \cdot \left[\sum_{j=0}^{m-k-MBS} p_j + \sum_{i=m-k-MBS+1}^{m-k-1} p_i G(m - k - i) \right] \quad (23)$$

IV. Numerical Validation

The resulting balance equations of analytical model have been solved using Maple 13 from Maplesoft, Inc. [10]. To validate the analytical solution, we have built a discrete event simulator of the cloud server farm using object-oriented Petri net-based simulation engine Artifex by RSoftDesign, Inc. [11].

We have performed two experiments for the system with different burst sizes; different configurations of system, $M^{g(x)}/G/100/150$ with traffic intensity of $\rho = 0.85$ and $CoV = 0.5$, 1.4 respectively, have been examined. The probability distribution of task service time is assumed to be *Gamma* and the burst size's one is assumed to be Geometric with the mean burst sizes of $x = 1, 2, 4, 6$ and 8.

Gamma distribution is selected for task service time since the CoV of Gamma distribution could be set independently of mean; this issue is important to us because we are interested to analyse the behaviour of system with a hyper-exponential service time, $CoV = 1.4$, as well as a hypo-exponential service time, $CoV = 0.5$.

First we present mean number of tasks in the system and the results are presented in Fig. 5. As can be seen mean number of tasks in the system decreases smoothly while the burst size is getting larger. In case of large burst, some space in the system will remain unused; so the number of tasks in system decreases as burst size becomes larger. We have

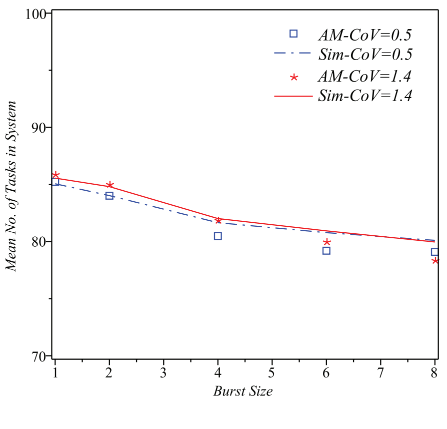

Fig. 5. Mean Number in System, $M^{g(x)}/G/100/150$, $\rho = 0.85$.

also computed the blocking probability and the probability of immediate service for all burst sizes. Blocking probabilities for system with different CoV are shown in Fig. 6. Results confirm

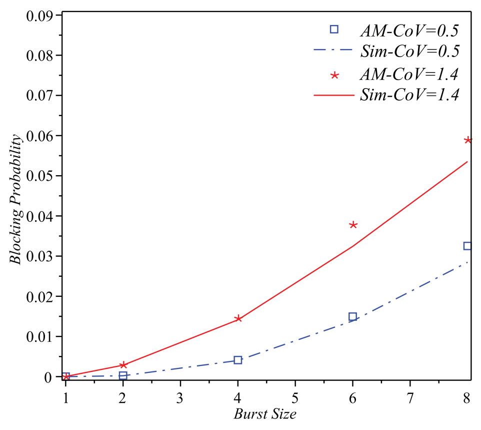

Fig. 6. Blocking Probability, $M^{g(x)}/G/100/150$, $\rho = 0.85$.

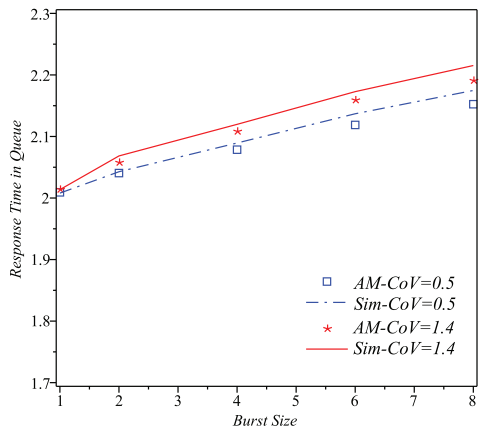

Fig. 8. Response Time in Queue, $M^{g(x)}/G/100/150$, $\rho = 0.85$.

that if the burst size increased linearly the blocking probability would also increase accordingly. Since the percentage of tasks which can get immediately into service is an important non-functional service property in SLA, we also demonstrate the probability of immediate service. As the Fig. 7 shows, like blocking probability, the increasing of burst size will decrease the probability of immediate into service linearly. Because of

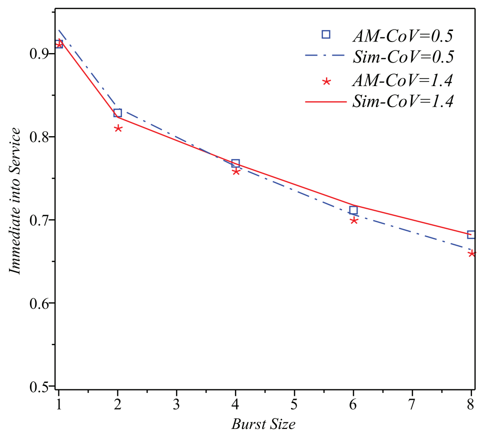

Fig. 7. Immediate into Service, $M^{g(x)}/G/100/150$, $\rho = 0.85$.

the total rejection policy, the big bursts will remain in the queue until all the required servers become idle; consequently the bigger the burst size is resulted in the longer the response time. Fig. 8 depicts the trend of response time while the burst size is changed.

As a general result, it can be seen that for cloud providers, maintaining SLAs is more difficult in case of having hyper-exponential distribution, $CoV = 1.4$, for service time. One potential way for reducing the risk of SLA violation could be the classification of task requests into different classes and taking care of them in separated queues; in such a scenario the distribution of task's service time would be a kind of hypo-exponential family (those with CoV less than one); provided that as our result shows (results for CoV = 0.5) supporting of SLA would be an easier task for cloud providers.

V. Conclusions

Maintaining the QoS at an acceptable level is of great importance aspect of cloud computing which is of crucial interest for both cloud providers and customers. According to our best knowledge, this paper is the first proposed analytical model for performance evaluation of a cloud computing center under burst arrival with total rejection policy. Due to the nature of the cloud environment, we assumed generally distributed service time for each task within super-tasks as well as large number of servers. We have further conducted numerical experiments and simulation to validate our model. Numerical and simulation results showed that the proposed method provided a quite accurate computation of the mean number of tasks in the system, mean response time, blocking probability and the probability of immediate service under burst arrival and total rejection policy for both admission and servicing.

References

[1] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, 2009.

[2] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *Proceedings of the 2009 Congress on Services - I*, 2009, pp. 693–700.

[3] L. Wang, G. V. Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, pp. 137–146, 2010.

[4] H. Khazaei, J. Mišić, and V. B. Mišić, "Modelling of cloud computing centers using M/G/m queues," *The first international workshop on Data Center Performance*, March 2011.

[5] D. D. Yao, "Some results for the queues $M^x/M/c$ and $GI^x/G/c$," *Operations Research Letters*, vol. 4, no. 2, pp. 79–83, 1985.

[6] A. Federgruen and L. Green, "An M/G/c queue in which the number of servers required is random," *Journal of Applied Probability*, vol. 21, no. 3, pp. 583–601, 1984.

[7] G. P. Cosmetatos, "Some practical considerations on multi-server queues with multiple poisson arrivals," *Omega*, vol. 6, no. 5, pp. 443–448, 1978.

[8] T. Kimura and T. Ohsone, "A diffusion approximation for an M/G/m queue with group arrivals," *Management Science*, vol. 30, no. 3, pp. 381–388, 1984.

[9] H. Takagi, *Queueing Analysis*. Amsterdam, The Netherlands: North-Holland, 1993, vol. 2: Finite Systems.

[10] Maplesoft, Inc., *Maple 13*, Waterloo, ON, Canada, 2009.

[11] RSoft Design, *Artifex v.4.4.2*. San Jose, CA: RSoft Design Group, Inc., 2003.