DDSS: Dynamic Dedicated Servers Scheduling for Multi Priority Level Classes in Cloud Computing

Husnu S. Narman*
husnu@ou.edu

Md. Shohrab Hossain†
mshohrabhossain@cse.buet.ac.bd

Mohammed Atiquzzaman*
atiq@ou.edu

*School of Computer Science, University of Oklahoma, Norman, OK 73019
†Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh

***Abstract*—Simplicity of usage, flexibility of data access, ease of maintenance, time and energy efficiency, and pay as you go policy have increased the usage of cloud computing over traditional computing. Cloud computing should be able to meet the performance expectations of different classes of customers. On the contrary, inefficient scheduling algorithms decrease the quality of service experienced by users. To improve quality of service, there are several proposed scheduling algorithms in the literature. However, these scheduling algorithms are limited because they do not consider different types of customers. Our objective is to satisfy performance expectations of customers by proposing an efficient Dynamic Dedicated Server Scheduling (DDSS) while considering different types of customers. Results show that the customer drop rate and throughput can be significantly improved by DDSS. Our proposed scheduling and related analysis will help cloud service providers build efficient cloud computing service architectures through considering different types of priority class performances such as, drop rate, throughput, and utilization.**

***Index Terms*—Cloud Computing, analytical modeling, homogeneous, multi server, multi class, queuing system.**

I. Introduction

Simplicity of usage, flexibility of data access, ease of maintenance, time and energy efficiency, and pay as you go policy have increased the usage of cloud computing over traditional computing [1], [2]. However, inefficient cloud server scheduling can lead to unwanted long delay and lower throughput.

The scheduling system needs to be suitable for different types of incoming service requests (from different classes of customers) [3], [4] because cloud computing systems can consist of different customer classes (such as, paid and unpaid customers). For example, expectations of paid customers are much higher than unpaid ones; this also implies some customer classes must have higher priority than others. Priority definition in cloud computing is different than the general definition of priority in queuing systems. In cloud computing, priority can be used to decide the next customer to be served and allocate the amount of resources for each customer class. Fig. 1 shows high and low priorities with and without priority levels. In cloud computing, low and high priority classes can be assigned to different priority levels and this can help in getting different quality of service from cloud servers. Therefore, some measure is needed to quantify the difference between high and low priorities. In addition, scheduling algorithms need to use resources (such as, services) efficiently. Therefore, the scheduling algorithm for cloud computing should efficiently fulfill desired expectations of different classes of customers without wasting resources [1].

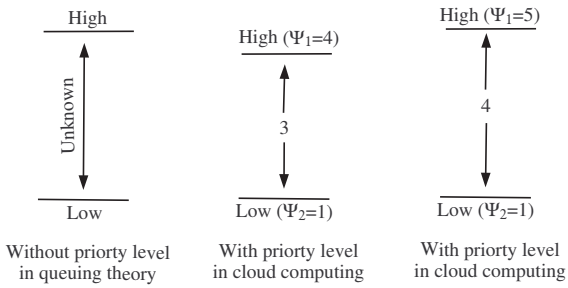

Fig. 1. High and Low class priorities without and with priority level.

Several research works [5]–[7] have been reported in the literature that proposed scheduling algorithms for cloud computing. Authors in [5], [7] proposed a queuing based analysis and scheduling model for performance evaluation of cloud systems using web applications as queues and virtual machines as service providers. Though creating unlimited virtual machines for each connection increases allocation rates, this dramatically decreases the performance of the system due to high response times [5] for each service request. Yang et al. [6] proposed a fault recovery system scheduling for cloud services and analyzed the system as an open queue problem. However, the result showed that addition of fault recovery increases average response times.

There have been a few research works [8]–[11] reported in the literature that analyzed the performance of cloud computing systems. Authors in [8], [11] evaluated the performance of several cloud service providers (such as, Amazon EC2, GoGrid) for scientific computing tasks, and found that the service providers are not ready to serve large data sets. Authors in [9], [10] used single class and single queue models to analyze the performance of cloud computing and found several performance distributions. However, none of the above works [5]–[11] considered the real case scenarios of cloud servers having different classes of customers (such as, unpaid and paid customers).

To the best of our knowledge, only Ellens et al. [3] and Hu et al. [4] proposed scheduling algorithms for cloud services having multiple customer classes. Hu et al. [4] used shared and Dedicated Server Scheduling (DSS) for two priority classes and obtained minimum number of servers to serve each class to satisfy certain performance. Ellens et al. [3] proposed hybrid scheduling that use two priority classes and reserves some servers for each class and shares remaining servers. However, these two works have not considered the class priority level difference as shown in Fig. 1. In this paper, we have proposed a novel scheduling algorithm that uses the priority level of classes, customer arrival rates, and number of services to dynamically update service rates (or number of dedicated servers) for each class of customers. We have also showed the impacts of the priority level of classes on the performance of the cloud computing system and compared our proposed Dynamic Dedicated Server Scheduling (DDSS) with DSS models.

The *objective* of this work is to improve performance of cloud systems in terms of throughput, drop rate, and utilization by considering class priority level. The *contributions* of this work are: (i) proposing Dynamic Dedicated Server Scheduling (DDSS) to fulfill the desired expectations for each level of priority class in the system, (ii) developing an analytical model to evaluate the performance (average occupancy, drop rate, average delay, and throughput for each class) of the proposed scheduling algorithm, (iii) validating our analytical model by an extensive simulation, and (iv) using class-based analysis to compare the class performances of DDSS and DSS.

Results show that the drop rate and throughput of customers can be significantly improved by using appropriate priority levels for classes in the proposed DDSS system. Our proposed realistic scheduling algorithm and related analysis will help cloud service providers build efficient cloud service architecture by considering different types of priority class performances, such as, drop rate, throughput, and utilization.

The rest of the paper is organized as follows. In Section II, we explain the typical DSS and proposed DDSS architectures. Section III presents the analytical model to derive different performance metrics of DDSS. In Section IV, we present the simulation and numerical results and compare the performances of DDSS and DSS systems. Finally, Section V has the concluding remarks.

II. Proposed Dynamic Dedicated Servers Architecture (DDSS)

Fig. 2 shows the DSS architecture for class 1 (C_1) and class 2 (C_2) customers. Here, some servers are used for C_1 customers while other servers are used for C_2 customers. The customer arrival rates of C_1 and C_2 are λ_1 and λ_2, respectively. Each class of customers are queued in the corresponding queues (Q_1 and Q_2). A new arriving customer will be dropped if buffers are full. The service rate of each server is μ. Each class of traffic is solely assigned to each dedicated server as shown in Fig. 2.

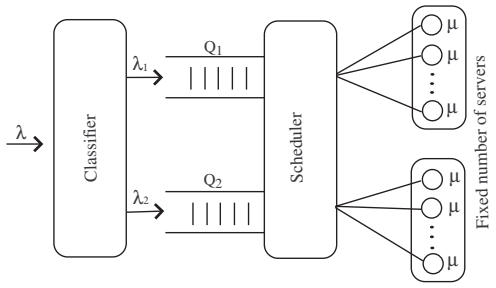

Fig. 2. Dedicated Servers Scheduling (DSS) Architecture.

In the typical DSS [3], [4], there is absolutely no sharing of traffic among dedicated servers, and the number of servers for each class is not updated dynamically. However, our proposed DDSS scheme (see Fig. 3) frequently updates the number of dedicated servers for each class according to their priority levels and arrival rates.

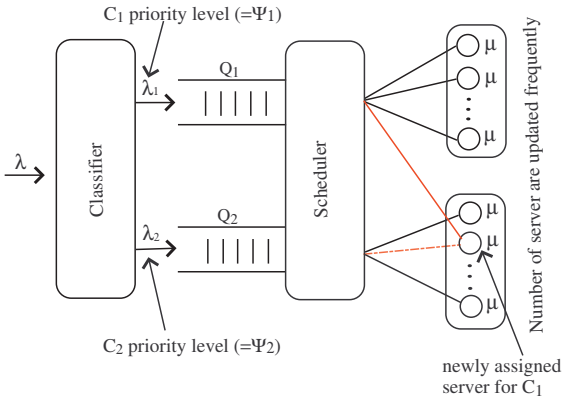

Fig. 3. Proposed Dynamic Dedicated Servers Scheduling (DDSS) Architecture.

A. Notations

The notations used in this paper are as follows:

p_i	Probability of i number of C_1 customers in the system,
λ	Arrival rate of customers,
λ_1,λ_2	Arrival rates of C_1 and C_2 customers, respectively,
Ψ_1,Ψ_2	Priority levels of C_1 and C_2 customers, respectively,
μ	Service rate of each server,
l	Total number of servers in the system,
m, k	Dedicated number of servers for C_1 and C_2,
N	Size of Q_1,
δ	Average delay of C_1 customers,
n	Average class occupancy of C_1 customers,
D	Drop probability of C_1 customers,
γ	Throughput of C_1 customers.

B. Scheduling Algorithm

Our proposed algorithm considers three crucial parameters that enable dynamic scheduling: (i) the arrival rates of C_1

and C_2 customers (λ_1, λ_2), (ii) the priority levels of C_1 and C_2 customers (Ψ_1, Ψ_2), and (iii) the total number of servers in the system (l). These three parameters are used to derive the number of servers (m and k) assigned to each class of customers as follows:

$$m = \left\lfloor \frac{l\Psi_1\lambda_1}{\Psi_1\lambda_1 + \Psi_2\lambda_2} \right\rfloor \tag{1}$$

$$k = l - m \tag{2}$$

$\Psi_2\lambda_2 \neq 0$ guarantees $k \neq 0$. For example, there are five servers in the system and two classes of customers having priority levels, $\Psi_1 = 5$ and $\Psi_2 = 3$, and arrival rates λ_1 = 20 and λ_2 = 10. By substituting these values in Eqns. (1) and (2), we get m = 4 and k = 1.

Eqn. (1) can easily be extended for an r multi-class system by assuming $\Psi_i\lambda_i \neq 0$ where $i = \{1, ...r\}$ as follows:

$$m_1 = \left\lfloor \frac{l\Psi_1\lambda_1}{\Psi_1\lambda_1 + \Psi_2\lambda_2 + \ldots + \Psi_r\lambda_r} \right\rfloor \tag{3}$$

Here, m_1 is the number of servers assigned to class 1. After finding m_1, remaining number of servers are $l_1 = l - m_1$. Hence, the number of servers assigned to class 2 can be obtained as follows:

$$m_2 = \left\lfloor \frac{l_1\Psi_2\lambda_2}{\Psi_2\lambda_2 + \Psi_3\lambda_3 + \ldots + \Psi_r\lambda_r} \right\rfloor \tag{4}$$

Iteratively following Eqns. (3) and (4), dedicated number of servers for each class can be measured.
The scheduling algorithm is as follows:

- Different class of customers are assigned to different servers.
- To avoid service degradation, servers simultaneously serve only limited number of customers.
- Number of assigned servers to each class is updated regularly based on the Eqns. (1) and (2).

When the scheduling algorithm computes the new values of m and k, some of the servers (which were previously serving C_2) will be assigned to C_1 (see Fig. 3). The servers will continue to serve C_2 customers until they finish their request. However, scheduling algorithm will not assign any new C_2 customer to the servers which are recently assigned to C_1. This strategy protects the customers (in service) from experiencing large delay or drop.

III. Analytical Model

The analytical model to derive different performance metrics of the DDSS architecture is presented in this section.

A. Assumptions

To make the model analytically tractable, it is assumed that the queuing system is under heavy traffic flows, customer arrivals follow Poisson distribution, and service times for customers are exponentially distributed. Type of queue discipline used in the analysis is FIFO. Service rate of all servers are equal (meaning the system is a homogeneous system).

The service rate of the system is state dependent. With one customer in the system, the service rate is μ; with two customers in the system, the service rate is 2μ. The service rate of the system increases until all the servers are utilized (m servers for C_1 customers and k servers for C_2 customers). Then the total service rate of the system is fixed at $m\mu$ and $k\mu$ (using Eqns. (1) and (2)) for C_1 and C_2 customers, respectively. Only C_1 customers performance metrics are derived in this section based on [12]. However, performance metrics (such as, average occupancy, average delay, drop rate, and throughput) of C_2 can be derived easily using k instead of m in Eqns. (7), (8), (11), and (12).

B. State Probabilities

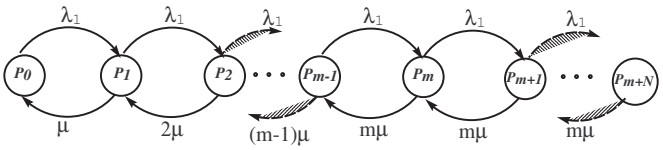

Fig. 4. State transition diagram for the model

Fig. 4 shows the state transaction diagram of the proposed DDSS model where p_i represents probability of i customers from C_1 in the system. λ_1 and $i\mu$ represent the state transition probabilities, where i = 1, 2, …, m. Based on the state transition diagram in Fig. 4, state probabilities can be formulated. D and γ can be computed for homogeneous multi-server system by using state probabilities [13]. In short, state probability equations can be written as follows by using M/M/c/N [12]–[14]:

$$p_i = \begin{cases} p_0 \frac{\rho^i}{i!} & , 1 \leq i \leq m \\ p_0 \frac{m^m}{m!}\rho_2^i & , m < i \leq m + N \end{cases} \tag{5}$$

Using $\sum_{i=0}^{m+N} p_i = 1$, we get

$$p_0^{-1} = \begin{cases} 1 + \sum_{i=1}^{m} \frac{\rho^i}{i!} + \frac{m^m}{m!} \sum_{i=m+1}^{m+N} \rho_2^i & , \rho_2 \neq 1 \\ 1 + \sum_{i=1}^{m} \frac{\rho^i}{i!} + N\frac{m^m}{m!} & , \rho_2 = 1 \end{cases} \tag{6}$$

where $\rho = \lambda_1/\mu$ and $\rho_2 = \lambda_1/m\mu$

C. Drop Probability and Throughput

The drop probability of the model is the final state probability which is p_{m+N}. Therefore, the drop rate and throughput of the model can be obtained as follows:

$$D = p_0 \frac{m^m}{m!}\rho_2^{m+N} \tag{7}$$

$$\gamma = \lambda_1(1 - D) \tag{8}$$

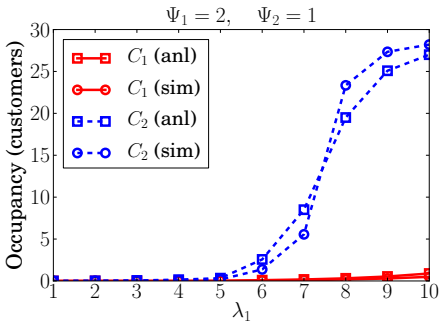

Fig. 5. Average class occupancy of DDSS obtained through simulations and analytical model.

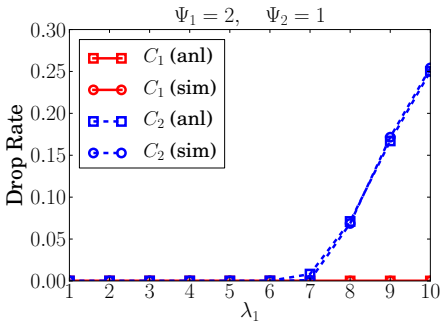

Fig. 6. Class drop rate of DDSS obtained through simulations and analytical model.

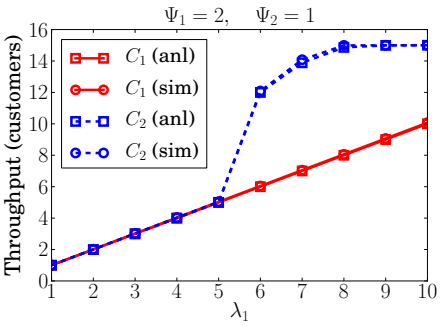

Fig. 7. Class throughput of DDSS obtained through simulations and analytical model.

D. Average Class Occupancy and Delay

The average class occupancy and average delay can be formulated by using state probabilities [13]. The average class occupancy, (n) for $M/M/1/N$ queue is as follows:

$$n = \sum_{j=1}^{N} j p_j \tag{9}$$

However, $M/M/c/N$ queue system has m servers and from the above state probabilities (Eqn. (5)), n is given by

$$n = \sum_{j=m+1}^{m+N} (j-m) p_j \tag{10}$$

which gives the following expressions for n:

$$n = \begin{cases} p_0 \rho_2 \frac{m^m}{m!} \left(\frac{1-(N+1)\rho_2^N + N\rho_2^{N+1}}{(1-\rho_2)^2} \right) & \rho_2 \neq 1 \\ p_0 \frac{m^m}{m!} \left(\frac{N(N+1)}{2} \right) & \rho_2 = 1 \end{cases} \tag{11}$$

Using Little's law and Eqns. (8) and (11), the average delay can be obtained as follows:

$$\delta = \frac{n}{\gamma} \tag{12}$$

IV. Results

Discrete event simulation has been carried out under the assumptions and scheduling policies mentioned in Section II. We have followed M/M/c/N [13] procedures to implement the simulation. Each buffer has a capacity to hold only 30 customers and there are six servers, each having service rate of μ = 5. We ran each simulation with 20000 samples for 10 trials having different arrival rates and priority levels as follows:
$\lambda_1 = \{i, j\}$, $\lambda_2 = \{i, 2j\}$, where i = 1,2, ...,5 and j = 6,7, ...,10 and $\Psi_1 = \{1.5, 2, 5\}$, $\Psi_2 = \{1\}$.

A. Validation of Analytical Model

In this subsection, we show the analytical and simulation results of the proposed DDSS architecture and compare them to validate our analytical approach.

1) Average Occupancy: Fig. 5 shows the average class occupancy of DDSS obtained through simulations and analytical model. The simulation and analytical results are very close to each other. The average occupancies of C_1 and C_2 are very low upto $\lambda_1 = 5$ because of the low arrival rates of both classes. After that, although both classes are served by equal number of servers, occupancy of C_2 increases sharply. There are two reasons for this: (i) the arrival rate of C_2 is twice the arrival rate of C_1, and (ii) the priority level of C_1 is twice the priority level of C_2.

2) Drop Rate and Throughput: Figs. 6 and 7 show the drop rate and throughput, respectively for both classes. The analytical results closely match with the simulation results in both cases. The drop rate of C_2 is higher than the drop rate of C_1 after $\lambda_1 = 6$ because the priority level of C_2 is not high enough (while arrival rates of C_2 are high) to get more servers.

Thus, it is evident that the obtained analytical and simulation results in Figs. 5, 6, and 7 are close to each other, thereby validating our analytical model.

B. Effects of Priority Levels on Performance of Classes

In this subsection, we present the impact of class priority levels and arrival rates on the performance of classes in DDSS approach. Moreover, significance of priority levels of each class are compared by keeping the value of Ψ_2 fixed 1 while changing the value of Ψ_1 at 1.5, 2, and 5. DDSS and DSS can be compared with respect to two important aspects: (a) How does the priority levels of classes affect the performance of classes in the system? 2) How are the performances of classes affected when using dynamically updated services in DDSS vs. fixed services in DSS? In this paper, we have only compared the impacts of the class priority levels on the class performances. Therefore, we assume that the DSS can dynamically update assigned number of servers for each class based on arrival rate as is done in DDSS. However, in reality DSS cannot update the assigned number of servers [4]. The class priority levels in DSS are constant and are assumed to be $\Psi_1 = 2$ and $\Psi_2 = 1$. Therefore, we compare the impact of class priority level on the class performance with respect to utilization, drop rate, throughput, and occupancy.

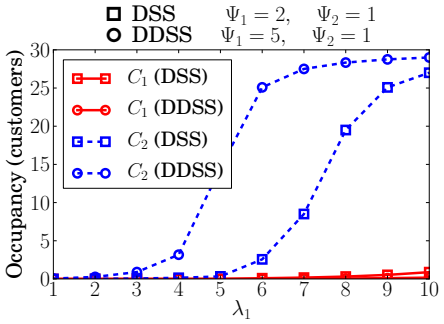

Fig. 8. Average class occupancies for DDSS and DSS.

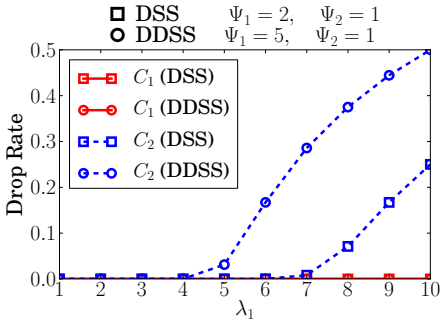

Fig. 9. Class drop rates for DDSS and DSS.

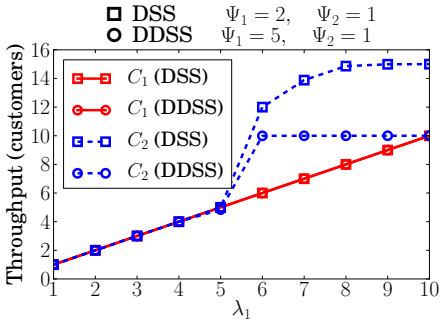

Fig. 10. Class throughput for DDSS and DSS.

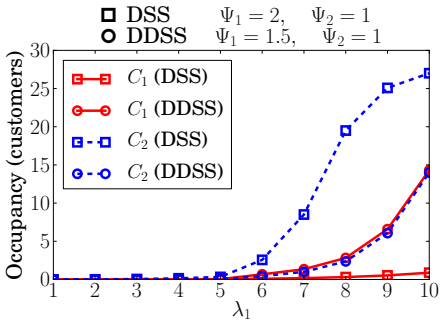

Fig. 11. Average class occupancies for DDSS and DSS.

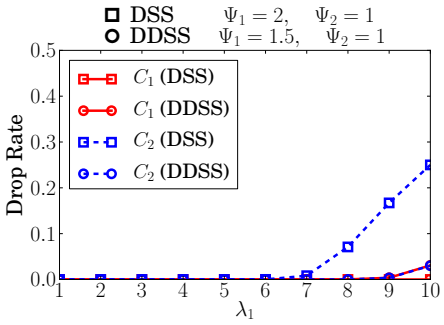

Fig. 12. Class drop rates for DDSS and DSS.

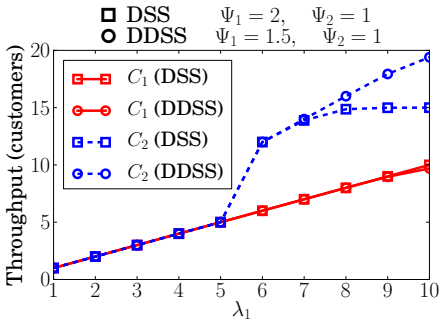

Fig. 13. Class throughput for DDSS and DSS.

1) Average Class Occupancy: Figs. 8 and 11 show the average occupancies for the proposed DDSS and DSS architectures. Fig. 8 shows that there is no significant difference in C_1 occupancies between DSS and DDSS although C_2 occupancy of DSS is much lower than C_2 occupancy of DDSS. This is because of fewer number of servers assigned to C_2 and larger number of servers assigned to C_1 in DDSS than DDS. As a result, C_2 suffers from large delay in DDSS over DSS. In addition, C_1 occupancy of DSS and DDSS are less than C_2 occupancy of DSS and DDSS (see Fig. 8).

In Fig. 11, we change the priority level of C_1 customers from $\Psi_1 = 5$ to $\Psi_1 = 1.5$ while keeping $\Psi_2 = 1$ (constant). We find that the results are reversed because DDSS have now flexibility on priority level.

The performance of DSS remains similar because of its non-flexibility on priority level. C_2 occupancy of DDSS is found to be significantly lower than C_2 of DSS although the occupancy of C_1 of DSS is lower than the occupancy of C_1 of DDSS (see Fig.11). It is worth mentioning that C_1 and C_2 occupancies of DDSS are almost same while there is larger difference between C_1 and C_2 occupancies of DSS. This is because of fewer number of servers assigned to C_1 in DDSS than DSS. If results in Figs. 8 and 11 are evaluated based on class priority level, the gap between the average occupancy of C_1 and C_2 get significantly larger while the difference between the priority level of C_1 and C_2 increases (see the difference between C_1 (DDSS) and C_2 (DDSS) in Fig. 8) and get close to each other, while the difference between the priority level of of C_1 and C_2 decreases (see the difference between C_1 (DDSS) and C_2 (DDSS) in Fig. 11).

2) Drop Rate and Throughput: Figs. 9 and 12 show the class drop rates for DDSS and DSS architectures. Fig. 9 shows that all of the class drop rates are very low in DSS and DDSS due to the low arrival rates upto $\lambda_1 = 5$. However, C_2 drop rate of DDSS is almost twice larger than C_2 drop rate of DSS after $\lambda_1 = 5$. In both DSS and DDSS, C_1 drop rate is lower than C_2 drop rate. These are because fewer number of servers are assigned to C_2 and larger number of servers being assigned to C_1 in DDSS than DSS (see Fig. 9). On the other hand, when we change Ψ_1 = 5 to $\Psi_1 = 1.5$ while keeping $\Psi_2 = 1$ constant (in Fig. 12), the drop rate of C_1 in DDSS is significantly reduced compared to the drop rate of C_2 in DSS after $\lambda_1 = 7$. If results in Figs. 9 and 12 are evaluated based on the class priority levels, the impact of C_1 priority levels on the drop rate is not realized. However, usage of different C_1 priority levels have significant impact on C_2 drop rate (see Fig. 9 for $\Psi_1 = 5$ and $\Psi_2 = 1$ and Fig. 12 for $\Psi_1 = 1.5$ and $\Psi_2 = 1$).

Figs. 10 and 13 show the class throughput for DDSS and DSS. Upto $\lambda_1 = 5$, the throughput of C_1 and C_2 for DSS and DDSS are similar since the assigned number of servers are enough to serve the related traffic. However, in Fig. 10, C_2 throughput of DSS is higher than that of DDSS (while there is no difference between C_1 throughput) when $\lambda_1 = 5, .., 10$. For C_1, the number of servers assigned to C_1 in DSS are enough to serve C_1 customers. On the other hand, Fig. 13 shows that C_2 throughput of DDSS is higher than C_2 throughput of DSS while the throughput of C_1 for DSS and DDSS are same (when Ψ_1 changes from 5 to 1.5 for DDSS). This shows how the priority level and DDSS can increase C_2 throughput of the

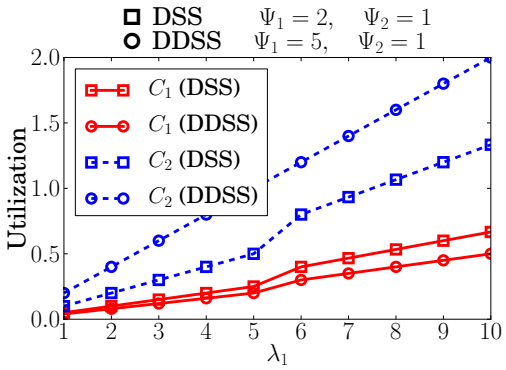

Fig. 14. Utilization of the classes for DDSS and DSS while $\Psi_1 = 5$ for DDSS.

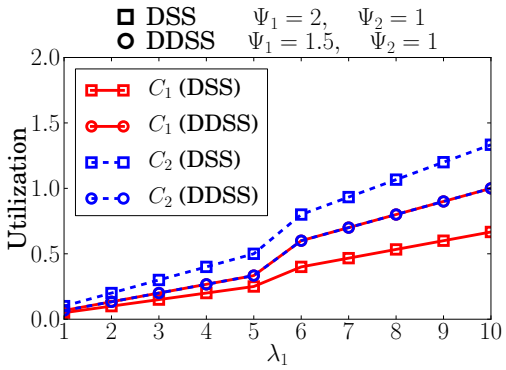

Fig. 15. Utilization of the classes for DDSS and DSS while $\Psi_1 = 1.5$ for DDSS.

system without affecting the other class.

3) Utilization: Utilization is a performance measure which reflects the efficiency of server usage. Here, the utilization is computed as the ratio of incoming class arrival rates to the total service rates (of all the dedicated servers) for the same class. We are interested in the impact of changes in the priority levels on the utilization of the system. Figs. 14 and 15 show the utilization of two classes for DSS and DDSS architectures. The gap between utilization values are higher in Figs. 14 (where the difference of the priority levels of two classes is 1) compared to the gaps in Fig. 15. The utilization gap between C_1 and C_2 in Fig. 15 is almost zero (where where the difference of the priority levels of two classes is 0.5). It is worth mentioning that increased C_1 priority level reduces C_1 utilization while improving C_2 utilization due to low C_1 arrival rates.

C. Summary of Results

Based on the results, we make the following observations: (i) the class priority levels do not significantly affect the performance of classes when the system is under low traffic for both DSS and DDSS architectures, (ii) under heavy traffic, the class priority levels have direct impact on the class performances in DDSS architecture, (iii) the system can become more efficient based on the selected class priority levels in DDSS than DSS, although assuming DSS can dynamically update the assigned number of servers for each class based on arrival rates ($\Psi_1 = 1.5$ and $\Psi_2 = 1$ in Fig. 13).

V. Conclusion

In this paper, we have proposed a scheduling algorithm for cloud computing by considering priority between customer classes. Analytical formulations of the proposed Dynamic Dedicated Server Scheduling are presented through different cases of priority levels. Performances of different classes under realistic scenarios have been compared through extensive simulations. Results show that DDSS architecture has the capability to improve the class throughput and reduce the drop rate without decreasing the performance of the high priority class while using system resources efficiently through appropriate level of priority. Therefore, DDSS makes cloud computing systems more efficient. The results obtained in this paper will help cloud service providers build efficient cloud computing service architectures for different types of customers.

References

[1] W. Kim, "Cloud Computing: Today and Tomorrow," *Journal of Object Technology*, vol. 8, pp. 65–72, Jan 2009.

[2] L. Wang, G. Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, Apr 2010.

[3] W. Ellens, M. Zivkovic, J. Akkerboom, R. Litjens, and H. van den Berg, "Performance of cloud computing centers with multiple priority classes," in *IEEE 5th International Conference on Cloud Computing (CLOUD)*, Honolulu, HI, June 24-29, 2012, pp. 245–252.

[4] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, "Resource provisioning for cloud computing," in *Conference of the Center for Advanced Studies on Collaborative Research (CASCON '09)*, Riverton, NJ, 2009, pp. 101–111.

[5] V. Goswami, S. S. Patra, and G. B. Mund, "Performance analysis of cloud with queue-dependent virtual machines," in *1st International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, Mar. 15-17, 2012, pp. 357–362.

[6] B. Yang, F. Tan, Y.-S. Dai, and S. Guo, "Performance evaluation of cloud service considering fault recovery," in *Cloud Computing*, Beijing, China, Dec. 1-4, 2009, pp. 571–576.

[7] H. peng Chen and S. chong Li, "A queueing-based model for performance management on cloud," in *6th International Conference on Advanced Information Management and Service (IMS)*, Seoul, Nov. 30-Dec. 2, 2010, pp. 83–88.

[8] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 931–945, June 2011.

[9] H. Khazaei, J. Misic, and V. Misic, "Performance analysis of cloud computing centers using M/G/m/m+r queuing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 936–943, May 2012.

[10] K. Xiong and H. G. Perros, "Service Performance and Analysis in Cloud Computing," in *IEEE Congress on Services*, Los Angeles, CA, July 6-10, 2009, pp. 693–700.

[11] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing," *Telecommunications Policy*, vol. 34, pp. 115–131, Oct 2010.

[12] F. S. Q. Alves, H. C. Yehia, L. A. C. Pedrosa, F. R. B. Cruz, and L. Kerbache, "Upper bounds on performance measures of heterogeneous M/M/c queues," *Mathematical Problems in Engineering*, vol. 2011, May 2011.

[13] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory (Wiley Series in Probability and Statistics)*. Wiley-Interscience, Feb 1998.

[14] H. Narman, M. S. Hossain, and M. Atiquzzaman, "Multi class traffic analysis of single and multi-band queuing system," in *IEEE Global Communications Conference (GLOBECOM)*, Atlanta, GA, Dec 9-13, 2013.