h-DDSS: Heterogeneous Dynamic Dedicated Servers Scheduling 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. However, inefficient scheduling algorithms decrease the quality of service experienced by users. To improve quality of service, several scheduling algorithms have been proposed in the literature. However, these scheduling algorithms do not consider heterogeneous servers or different priority classes of customers. Our objective is to satisfy performance expectations of customers by proposing a Heterogeneous Dynamic Dedicated Server Scheduling (h-DDSS) while considering heterogeneous servers and different priority classes of customers. Results show that h-DDSS architecture can provide improved customer throughput and drop rate over homogeneous DDSS in cloud computing. Our proposed scheduling algorithm and related analysis will help cloud service providers build efficient cloud service architectures which are adaptable to homogeneous and heterogeneous environments by considering different types of priority class performances, such as, drop rate, throughput, and utilization.**

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

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 system scheduling can lead to unwanted long delay and lower throughput.

Cloud server systems consist of heterogeneous servers (which means service rate of all servers can be different) because failed or misbehaved servers of a multi-server system are replaced by new and more powerful ones that cause systems to be heterogeneous [3]. Therefore, heterogeneous servers should be considered while designing an architecture for scheduling in cloud computing. Because of heterogeneity, the order of incoming customers distribution to servers (namely allocation policy, such as Fastest Server First (FSF), Slowest Server First (SSF)) also needs to be taken into account. In addition, the scheduling algorithms also need to be suitable for different class of customers [4], [5].

Cloud computing systems can consist of different customer classes, such as paid and unpaid customers; the expectations of these classes can be different. For example, expectations of paid customers are naturally much higher than unpaid ones; this means 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

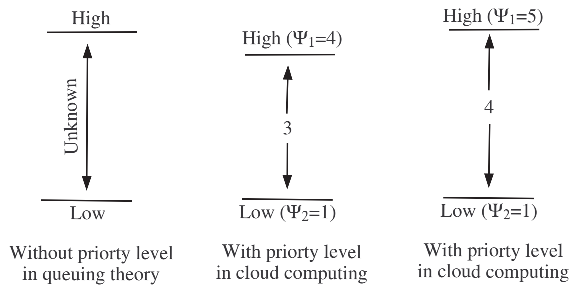
High
Unknown
Low
Without priorty level in queuing theory
High (Ψ_1=4)
3
Low (Ψ_2=1)
With priorty level in cloud computing
High (Ψ_1=5)
4
Low (Ψ_2=1)
With priorty level in cloud computing

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

with and without priority levels. As explained in our previous work [6], low and high priority classes can be assigned to different priority levels in cloud computing. 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. Moreover, scheduling algorithms need to use resources (such as, services) efficiently. Therefore, the scheduling algorithm for cloud computing should efficiently satisfy expectation of each different class of customers in heterogeneous environment without wasting resources [1].

Several research works [7]–[9] have reported scheduling algorithms for cloud computing. Authors in [7], [9] proposed a queuing based analysis and scheduling model for the 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 rate, this dramatically decreases the performance of the system due to high response time [7] for each service request. Yang et al. [8] proposed a fault recovery scheduling algorithm for cloud services and analyzed the system as an open queue problem. However, the result showed that addition of fault recovery increases average response time.

There have been a few research works [10]–[13] reported in the literature that analyzed the performance of cloud computing systems. Authors in [10], [13] 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 [11], [12] used single class and single queue models to analyze the performance of cloud computing and found several performance distributions. However, the *limitations* of the above works [7]–[13] are: (i) different classes of customers (such as, unpaid and paid customers) is not considered, and (ii) usage of heterogeneous servers is not taken into account in cloud networks. Therefore, the real case scenario of cloud server systems is not represented.

To the best of our knowledge, only Ellens et al. [4] and Hu et al. [5] proposed scheduling algorithms for cloud services having multiple customer classes. Hu et al. [5] 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. [4] proposed a hybrid scheduling that use two priority classes and reserves some servers for each class and shares remaining servers. However, these two works have considered neither class priority level nor heterogeneous server systems which require much harder complex analysis because of different allocation policies. Moreover, Narman et al. [6] is the only previous work exists that proposed a scheduling algorithm which taking into account the priority level of customers for cloud servers.

In [6], we have proposed DDSS for homogeneous server systems (which means service rate of all servers are equal) and compared the existing homogeneous DSS scheduling with proposed DDSS scheduling. However, it is essential to analyze performance of cloud computing in heterogeneous server systems and compare homogeneous server systems with heterogeneous server systems by considering different allocation policies (such as Fastest Server First (FSF), Slowest Server First (SSF) for heterogeneous servers).

In this paper, we have proposed a novel scheduling algorithm by considering heterogeneous servers, priority level of classes, customer arrival rates, and service rates of servers to dynamically update assigned service rates (or number of dedicated servers) to each class of customer. We have also showed the impact of the priority level of classes on the performance of the cloud computing systems which use Heterogeneous Dynamic Dedicated Server Scheduling (h-DDSS) and DDSS. Consequently, not only more realistic scenario but also performance difference between heterogeneous and homogeneous systems in cloud computing are analyzed and presented.

The *objective* of this work is to improve and realistically analyze performance of cloud systems in terms of throughput, drop rate, and utilization by considering heterogeneous servers and different priority class of customers. The *contributions* of this work are: (i) proposing h-DDSS to fulfill desired expectations for each level of priority class in the system, (ii) developing an analytical model to evaluate the upper and lower bounds performances (average occupancy, drop rate, average delay, and throughput for each class) of the proposed scheduling algorithm, (iii) validating our analytical model by extensive simulations, and (iv) comparing the performances of DDSS with upper and lower bounds performances of h-DDSS. *Results* show that h-DDSS architecture can provide improved customer throughput and drop rate over DDSS by using appropriate priority levels for each class.

The rest of the paper is organized as follows. In Section I, we explain typical DSS, DDSS [6], and proposed h-DDSS architectures. Section II presents analysis of the model to derive upper and lower bounds performance of h-DDSS. In Section III, we present the simulation and numerical results and comparing the performances of h-DDSS and DDSS. Finally, Section IV has the concluding remarks.

Proposed Heterogeneous Dynamic Dedicated Servers Architecture (h-DDSS)

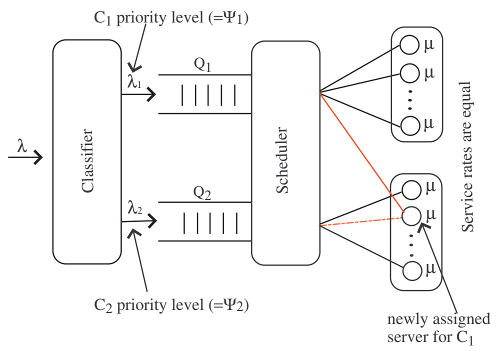

Fig. 2. Dynamic Dedicated Servers Scheduling (DDSS) Architecture.

DSS [4], [5] architecture for class 1 (C_1) and class 2 (C_2) customers is similar to Fig. 2. In DSS, 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). Arriving customers will be dropped if the buffers are full. μ is service rate of each server. Each class of traffic is solely assigned to each dedicated server and there is absolutely no sharing of traffic among the dedicated servers, and the number of servers for each class is not updated dynamically. However, DDSS scheme (see Fig. 2) frequently updates the number of dedicated servers for each class according to priority levels and arrival rates [6]. However, service rates of all servers are equal.

A. Notations

The notations used in the rest of the explanation and analysis are listed below.

p_i	Probability of i number of C_1 customers in the system,
λ	Arrival rate of customers,

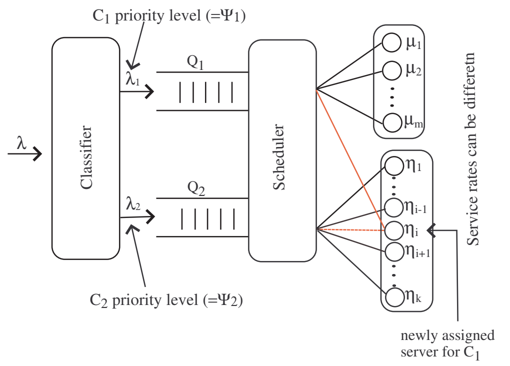

Fig. 3. Proposed Heterogeneous Dynamic Dedicated Servers Scheduling (h-DDSS) Architecture.

λ_1,λ_2	Arrival rates of C_1 and C_2 customers,
Ψ_1,Ψ_2	Priority level of C_1 and C_2 customers,
μ_i, η_i	Initial service rate of dedicated i servers for C_1 and C_2 customers,
μ_{ti}, η_{ti}	Sum of Service rate of dedicated servers for C_1 and C_2 customers until i^{th} server,
μ_{total}	Total service rates in the system,
m, k	Dedicated number of servers for C_1 and C_2,
N	Size of Q_1,
δ	Average queuing delay of C_1 customers,
n	Average queue 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 enables a 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 service rates in the system (μ_{total}). These three parameters can be used to derive total service rates (μ_{tm} and η_{tk}) assigned to each class of customers as follows:

$$\mu_{tm}^{'} = \left\lfloor \frac{\mu_{total}\Psi_1\lambda_1}{\Psi_1\lambda_1 + \Psi_2\lambda_2} \right\rfloor \tag{1}$$

Indeed, $\mu_{tm}^{'}$ is a ideal total service rate assigned to C_1. However, it is hard to achieve equaling sum of heterogeneous service rates to $\mu_{tm'}$. Therefore, the faster servers (sum of servers of these servers are approximately equal to μ_{tm}^{i}) are assigned to the higher priority class.

$$\mu_{tm} = \sum_{i=1}^{m} \mu_i \approx \mu_{tm}^{'} \tag{2}$$

$$\eta_{tk} = \mu_{total} - \mu_{tm} \tag{3}$$

$\Psi_2\lambda_2 \neq 0$ guarantees $k \neq 0$.

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

$$\mu_{tm1}^{'} = \left\lfloor \frac{\mu_{total}\Psi_1\lambda_1}{\Psi_1\lambda_1 + \Psi_2\lambda_2 + \ldots + \Psi_r\lambda_r} \right\rfloor \tag{4}$$

Here, μ_{tm1} ($= \sum_{i=1}^{m1} \mu_i \approx \mu_{tm1}^{'}$) is the total service rates assigned to C_1. After finding μ_{tm1}, the remaining service rates of servers are $\mu_{total}^{'} = \mu_{total} - \mu_{tm1}$. Hence, the total service rate assigned to C_2 can be obtained as follows:

$$\mu_{tm2}^{'} = \left\lfloor \frac{\mu_{total}^{'}\Psi_2\lambda_2}{\Psi_2\lambda_2 + \Psi_3\lambda_3 + \ldots + \Psi_r\lambda_r} \right\rfloor \tag{5}$$

Iteratively following Eqns. (4) and (5), the dedicated number of servers for each class can be measured.

The scheduling algorithm is as follows:

- Different classes 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. (2) and (3).

When the scheduling algorithm computes the new values of μ_{tm} and η_{tk}, some of the servers (which were previously serving C_2) will be assigned to C_1 (see Figs. 3 and 2). The servers will continue to serve C_2 customers until they finish their request. However, the 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.

II. Analysis

The analytical model to derive various performance metrics of h-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 can be different (meaning the system is heterogeneous). Because of heterogeneity, different allocation policies can be used. In analysis, FSF and SSF allocation policies are used to analyze the best and worst performances of h-DDSS. The best performance (upper bound) of the heterogeneous systems is obtained by using FSF allocation and the worst performance (lower bound) of the heterogeneous systems is obtained by using SSF allocation because arrived customer request can be served faster at servers which have higher service rates and slower at servers which have lower service rates.

The service rate of the system is a state dependent. With one customer is in the system, the service rate is $\mu_{t1} = \mu_1$ and when two customers are in the system, the service rate is $\mu_{t2} = \mu_1 + \mu_2$. Service rate of the system increases until all servers are utilized (m servers for C_1 customers and k servers for C_2 customers). Then the total server rate of the system is fixed at μ_{tm} and η_{tk} (using Eqns. (2) and (3)) for C_1 and C_2 customers, respectively. Only C_1 customers performance metrics are developed based on [14] because the metrics (average occupancy, average delay, drop rate, and throughput) of C_2 can be derived similarly.

B. State Probability

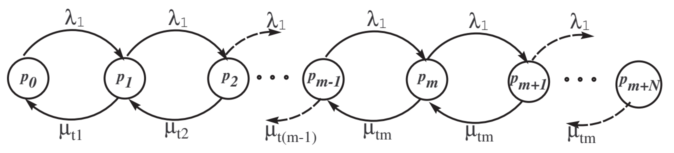

Fig. 4. State transition diagram for the model

Fig. 4 shows the state transaction diagram of the proposed model. p_i represents the probability of i customers from C_1 in the system and λ_1 and μ_{ti} represent the probabilities of state transition where i = 1, 2, ..., m. Based on the state transition diagram in Fig. 4, state probabilities can be formulated. Assuming $\mu_1 \leq \mu_2 \ldots \leq \mu_m$ gives class performance metrics of SSF allocation policy in h-DDSS. On the contrary, assuming $\mu_1 \geq \mu_2 \ldots \geq \mu_m$ gives class performance metrics of FSF allocation policy in h-DDSS. Thus, by using the state diagram of Fig. 4, the lower and upper bounds of class performance metrics, D, γ, n, and δ, are be computed for a heterogeneous multi-server system [15]. In short, state probability equations can be written as follows by using $M/M_i/c/N$ [14]–[16]:

$$p_i = \begin{cases} p_0 \frac{\lambda_1^i}{\prod_{j=1}^{i} \mu_{tj}} & i \leq m \\ p_0 \frac{\mu_{tm}^m \rho^i}{\prod_{j=1}^{m} \mu_{tj}} & m < i \leq m+N \end{cases} \tag{6}$$

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

$$p_0^{-1} = \begin{cases} 1 + \sum_{i=1}^{m} \frac{\lambda^i}{\prod_{j=1}^{i} \mu_{tj}} + \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}} \sum_{i=m+1}^{m+N} \rho^i & \rho \neq 1 \\ 1 + \sum_{i=1}^{m} \frac{\lambda^i}{\prod_{j=1}^{i} \mu_{tj}} + N \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}} & \rho = 1 \end{cases} \tag{7}$$

where $\rho = \lambda_1/\mu_{tm}$.

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:

$$D = p_0 \frac{\mu_{tm}^{m+N} \rho^{m+N}}{\prod_{j=1}^{m} \mu_{tj}} \tag{8}$$

$$\gamma = \lambda_1(1 - D) \tag{9}$$

D. Average Class Occupancy and Delay

The average class occupancy and average delay can be formulated by using state probabilities [15]. The average class occupancy, (n) for $M/M/1/N$ queue is as follows:

$$n = \sum_{j=1}^{N} j p_j \tag{10}$$

However, $M/M_i/m/N$ queue system has m servers and from the above state probabilities (Eqn. (6)), n is given by

$$n = \sum_{j=m+1}^{m+N} (j - m) p_j \tag{11}$$

which gives the following expressions for n:

$$n = \begin{cases} p_0 \frac{\mu_{tm}^m}{\prod_{i=1}^{m} \mu_{ti}} \rho^{m+1} \left(\frac{1-(N+1)\rho^N + N\rho^{N+1}}{(1-\rho)^2} \right) & \rho \neq 1 \\ p_0 \frac{\mu_{tm}^m}{\prod_{i=1}^{m} \mu_{ti}} \left(\frac{N(N+1)}{2} \right) & \rho = 1 \end{cases} \tag{12}$$

Using Little's law and Eqns. (9) and (12), the average delay can be obtained as follows:

$$\delta = \frac{n}{\gamma} \tag{13}$$

III. Results

Discrete event simulation has been carried out under the assumptions and scheduling policies mentioned in Section I. We have followed $M/M_i/c/N$ [15] procedures to implement the simulation. Each buffer has a capacity to hold only 30 customers and there are seven servers, each having different service rates (μ = 1, 2, ...7). We ran each simulation with 20000 samples for 10 trials having different arrival rates and priority levels as follows:
λ_1 = $\{i\}$, λ_2 = $\{2\lambda_1\}$, where i = 1,2, ...,10 and Ψ_1 = {2, 3}, Ψ_2 = {1}.

A. Validation of Analytical Model

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

1) Average Occupancy: Fig. 5 shows the average class occupancy of h-DDSS (for FSF allocation) obtained through simulations and analytical model. The simulation and analytical results are close to each other. Because of the low arrival rates of both classes, the average occupancies of C_1 and C_2 are low upto $\lambda_1 = 5$. After that, both classes are served by approximately equal service rates because of $\Psi_1 = 2$ and $\Psi_2 = 1$ while $\lambda_1 = i$ and $\lambda_2 = 2i$ (i = 6,...10.). Therefore, the occupancy of C_2 increases sharply because of higher arrival rates of C_2 customers.

2) Drop Rate and Throughput: Figs. 6 and 7 show the drop rate and throughput, respectively for both classes of h-DDSS obtained through simulations and analytical model. The analytical results exactly match with the simulation results for both the drop rate and throughput. 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 the arrival rates of C_2 are higher) to get more service rates.

As showed in Figs. 5, 6, and 7, the obtained analytical and simulation results are close to each other, thereby validating our analytical model.

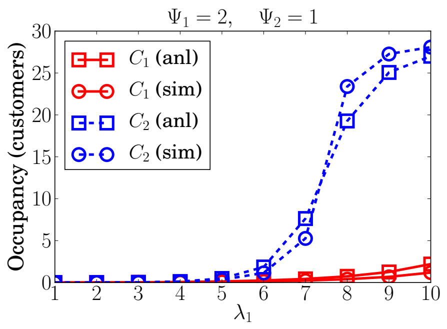

Fig. 5. Average class occupancy of h-DDSS (FSF allocation) obtained through simulations and analytical model.

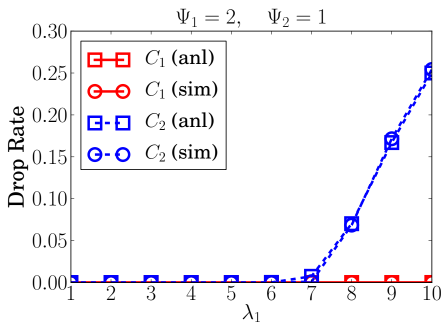

Fig. 6. Class drop rate of h-DDSS (FSF allocation) obtained through simulations and analytical model.

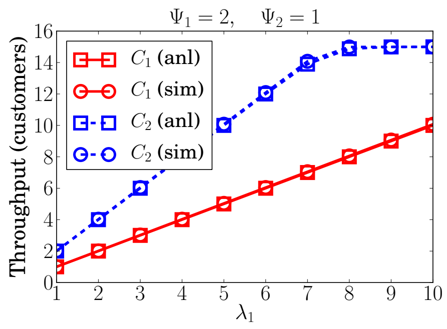

Fig. 7. Class throughput of h-DDSS (FSF allocation) obtained through simulations and analytical model.

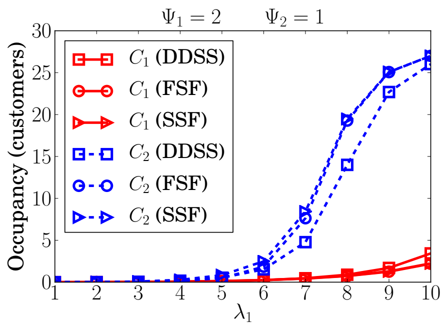

Fig. 8. Average class occupancies for DDSS and h-DDSS with FSF and SSF.

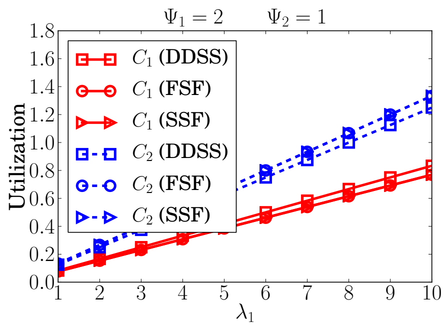

Fig. 9. Utilization of the classes for DDSS and h-DDSS with FSF and SSF.

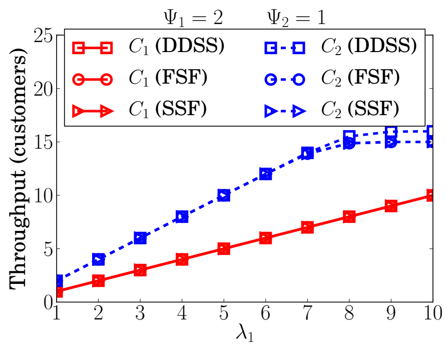

Fig. 10. Class throughput for DDSS and h-DDSS with FSF and SSF.

B. Performances of Classes in DDSS and h-DDSS

In this subsection, we present the difference between DDSS and h-DDSS by showing the performance of classes. DDSS results are obtained by averaging service rates of heterogeneous servers while keeping every other parameters same. For example, if total service rates of seven heterogeneous servers is equal 28, average service rates of each of the seven homogeneous servers is four. Moreover, significance of priority levels of each class are compared by keeping the value of Ψ_2 fixed at 1 while changing the value of Ψ_1 from 2 to 3. We can also compare h-DDSS with the typical DSS. However, impacts of the class priority levels on the class performance of DDSS and typical DSS are compared in our past work [6]. Therefore, we will not compare the typical DSS and h-DDSS is in this paper.

1) Average Class Occupancy: Figs. 8 and 11 show the average class occupancies for DDSS and h-DDSS architectures. Fig. 8 shows that C_1 and C_2 occupancies are low until $\lambda_1 = 5$ for all cases. While increasing the arrival rates, C_1 occupancy of DDSS is higher than C_1 occupancies of h-DDSS for both FSF and SSF allocations although C_2 occupancy of DDSS is lower than C_2 occupancies of h-DDSS for both FSF and SSF allocations. In Fig. 11, we change the priority of C_1 customers from $\Psi_1 = 2$ to $\Psi_1 = 3$ while keeping $\Psi_2 = 1$ (constant). We find that the results are reversed because DDSS is affected by priority levels of classes more than h-DDSS for both FSF and SSF allocations. When both Figs. 8 and 11 are considered together, the occupancy of C_1 and C_2 for both FSF and SSF of h-DDSS are similar while the system is under heavy customer traffic. However, under low customer traffic, it is expected that the occupancy of C_1 and C_2 for FSF is lower than SSF's ones.

2) Throughput: Figs. 10 and 13 show the class throughput for DDSS and h-DDSS architectures. Upto $\lambda_1 = 6$, C_1 and C_2 throughput for DDSS, SSF, and FSF are similar since the assigned number of servers is enough to serve related traffic. However, in Fig. 13, C_2 throughput of DDSS is lower than that of SSF and FFS (while there is no difference between C_1 throughput) after $\lambda_1 = 6$. On the other hand, there is no significant difference between C_2 throughput of DDSS and h-DDSS (SSF and FFS) when $\Psi_1 = 2$ (see Fig. 10). This is the result of priority level because when priority level gap between classes is higher, it reserves more servers for high priority 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 priority levels on the utilization of DDSS and h-DDSS. Figs. 9 and 12 show the class utilization for DDSS and h-DDSS architectures. The gap between utilization values of C_1 and C_2 of DDSS are lower than the utilization values of C_1 and C_2 of h-DDSS (FSF and

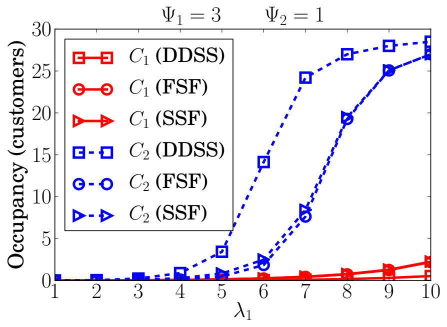

Fig. 11. Average class occupancies for DDSS and h-DDSS with FSF and SSF.

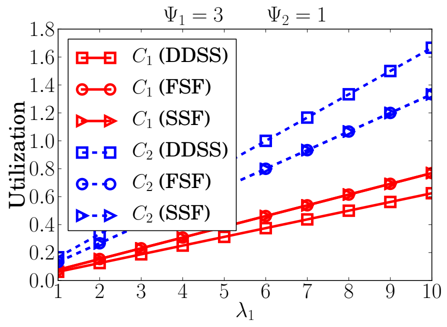

Fig. 12. Utilization of the classes for DDSS and h-DDSS with FSF and SSF.

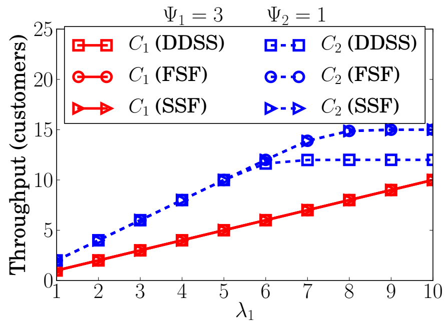

Fig. 13. Class throughput for DDSS and h-DDSS with FSF and SSF.

SSF) in Fig. 9 (where the difference of class priority level is low) compared to the gap in Fig. 12. It is worth mentioning that increasing C_1 priority level reduces C_1 utilization while improving C_2 utilization of DDSS.

C. Summary of Results

Based on the results, we make the following observations: (i) class priority levels do not significantly affect the performance of classes when system is under low traffic for both homogeneous and heterogeneous server systems, (ii) under heavy traffic, class priority levels have a significant impact on the class performances in homogeneous server systems, (iii) under heavy traffic, class performances of h-DDSS for FSF and SSF are same while FSF is better for low traffic arrivals, and (iv) heterogeneous systems can become as efficient as or better than homogeneous systems for selected class priority levels (see Figs. 10 and 13).

IV. Conclusion

In this paper, we have proposed a realistic scheduling algorithm for cloud computing by considering heterogeneous servers with priority classes. Expressions for upper and lower bounds performances of the heterogeneous Dynamic Dedicated Server Scheduling are presented through different cases of priority levels. Performance of different classes have been compared through extensive simulations for homogeneous and heterogeneous systems. Results show that heterogeneous systems, which use system resources efficiently through appropriate priority levels for classes, can provide improved customer throughput and drop rate over homogeneous systems in cloud computing. Our proposed scheduling algorithm and related analysis will help cloud service providers build efficient cloud service architectures which are adaptable to homogeneous and heterogeneous server systems by considering different types of priority class performances, such as, drop rate, throughput, and utilization.

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] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini, “Energy conservation in heterogeneous server clusters,” in *Principles and Practice of Parallel Programming*, Chicago, IL, June 15-17, 2005, pp. 186–195.

[4] 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.

[5] 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.

[6] H. Narman, M. S. Hossain, and M. Atiquzzaman, “DDSS:Dynamic Dedicated Servers Scheduling for multi priority level classes in cloud servers,” in *IEEE International Conference on Communications (ICC)*, Sydney, Australia, June 10-14, 2014.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.