JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 759-772 (2017)
DOI: 10.6688/JISE.2017.33.3.10

An Enhanced Priority Scheduling Algorithm for Multi-Server Retrieval Cloud System

G. NALINIPRIYA[1], K. G. MAHESWARI[2] AND K. KOTTESWARI[3]
[1]*Department of Information Technology*
Saveetha Engineering College
[2]*Department of Master of Computer Applications*
Institute of Road and Transport Technology
[3]*Department of Computer Science and Engineering*
Annai Mira College of Engineering and Institute
Anna University
Chennai, 602105 India

Cloud computing has become a most used business jargon in the area of high-performance computing as it provides on-demand services over ashared pool of internet resources. Process scheduling is one of the most important research issues to be focused on improving the performance of a cloud-based queuing system. The process of scheduling is to assign an appropriate resource to the task to achieve one or more objectives. Nowadays, the scheduling of jobs in cloud leads to NP-hard problem due to its large solution space and longer time to find an optimal solution. To find an optimal solution for the above issues and to improve theperformance of execution, a new priority based scheduling algorithm Enhanced Priority Scheduling Algorithm (EPSA) is proposed. The EPSA algorithm prioritizes jobs based on the job attribute. The attribute of jobsis execution time, resource requirement, CPU utilization, and dependency and target time. These parametersare combined and formulated for priority calculation. The job priority measure is used to generate a fuzzy decision tree (FDT). The FDT predicts the ordering position and service acquiring time for jobs that enter the cloud-based queuing system. EPSA algorithm improves the execution time and shows better performance compared to existing scheduling algorithms. The Proposed algorithm is implemented using JSIMgraph of JMT for performance modeling and evaluation.

Keywords: queuing model, scheduling, prioritization, fuzzy decision tree, performance

1. INTRODUCTION

Scheduling a job to a server and allocating the required resource for the execution of the job within finite duration is a significant process in the cloud environment, essential to guarantee the Quality of Service (QoS) [1]. Recently the cloud computing paradigm had introduced a High-Performance Computing (HPC) application that can run in the cloud in a cost-efficient and easy-to-use way. Amazon's Infrastructure as a Service solution, Amazon EC2 provides HPC instances for scheduling jobs [2]. As the paper is dealing with Scheduling algorithms, the prominent and widely used job scheduling algorithms like First Come First Services [3] and EASY (Extensible Argonne Scheduling sYstem) method [4] are studied. In these algorithms, the user submits the jobs to the waiting queue of the processor to run the jobs physically. EASY algorithm schedules the

Received July 16, 2016; revised August 31, 2016; accepted September 1, 2016.
Communicated by Balamurugan Balusamy.

jobs based on the Job's processor number and expected execution time. The estimation of expected running time of the job is difficult to predict in the cloud. For EASY, over- estimation may lead to longer waiting time and loss in CPU quota while under-estimation will lead to termination of jobs before the execution starts. The FCFS schedules the job only based on the processor number and doesn't consider the running time of the job; this may lead to severe fragmentation problem. Moreover, both FCFS and EASY doesn't consider the idle cycle of CPU while scheduling the jobs for execution.

In general, the NP-hard problem occurs while mapping tasks on unlimited cloud resources. Many scheduling algorithms are developed based on metaheuristic techniques. This Metaheuristic technique finds an optimal solution for NP-hard problems. The algorithms [5] based on this technique are Ant Colony Optimization (ACO), Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and BAT algorithm. Further scheduling varies based on the dependency of the task to be scheduled. In a job or task, if there is any existing order of execution, then the task can be executed, after the execution of parent task whereas independent task can be ordered and executed anywhere in the scheduling queue. The latter is called as Independent scheduling [7], and former is known as the dependent task or workflow scheduling [6]. In this paper, both types of scheduling methodology are considered for the scheduling process.

In this paper, we have discussed the problems of scheduling and attributes to be considered while scheduling jobs. An algorithm named Enhanced Priority scheduling algorithm (EPSA) is designed to overcome the problems and also to consider the necessary attributes of scheduling. EPSA is based on the priority of jobs. A Multi-Server Finite queuing system $M/M/c$ is constructed using the JSIMgraph tool [22] for imple- menting EPSA algorithm. This algorithm schedules the jobs in the $M/M/c$ system based on the priorities. The priority of each job is designed using a set of attributes of jobs, and the decision of which job to acquire the service first is taken using fuzzy logic. The attributes involved in prioritizing jobs are Expected Execution Time $E[T]$, Target time of the job T_T, Time to Live (TTL), CPU Utilization C_U, Dependency Ω, Independency of the task, and resource requirement of the jobs R_R. From the outcome generated, a fuzzy decision-making tree is drawn to arrange the jobs in an order of execution. The EPSA scheduling solves the NP-hard problems and fragmentation problems. The system performance and the execution time of the jobs can be improved using the EPSA algorithm. The main idea and contribution of the paper are to

- Develop a priority scheduling for priority based jobs
- To improves Quality of Service (QoS) and guarantees Service Level Agreement (SLA) by balancing and scheduling a task among jobs
- To find an optimal solution for NP-hard problem
- To improves execution time and waiting time of the queuing system
- To predict the job's ordering position and service acquiring time using fuzzy logic

The remaining part of paper is presented as follows: Section 2 surveys the related of the scheduling algorithm. Section 3 explains the proposed Priority based scheduling algorithm. Section 4 compare and analysis the performance of proposed algorithm using JSIMgraph tool. Section 5 concluded notifying the advantage in using proposed algorithm.

2. RELATED WORKS

In the computer system, many jobs share resources such as CPU, disks, and other devices. The jobs are executed one at a time, and the remaining jobs wait in the queue for resources. In the queuing system, there are two types of queues such as open and closed. In the open queuing system, the jobs enter the source and departure at the sink. The job arrival and departure vary with time. In closed network, there is no external arrival and departure. A constant amount of jobs keeps circulating from one queue to another. The source and sink are connected. The FCFS [8] is the most basic method of scheduling that is an open type. In FCFS, the job's runtime is not required for making any decision and hence has the severe fragmentation problem. EASY [9] is another method of scheduling which gives priority to the job having less running time and a smaller number of processors. In [10], the author proposed extended based scheduling. The extended based scheduling has a testing time for all the jobs. During the testing period, the jobs characteristic is predicted. After the testing period, the jobs get ready for normal execution based on their character. The traditional scheduling algorithm mainly focuses on the competence of the whole system. The scheduling algorithm economically optimizes the resource by allocating data according to their requirement and pricing [21].

The author [11] obtains the response time of the distribution of the cloud system modeled on an *M/M/m* open network. Using the response time, they estimate the optimal level of services and relationship between maximum and a minimum number of tasks. In [12], the response distribution time is obtained for *M/M/m/m+r* closed model. Here the response time is used to determine the probability of waiting time of tasks in the queue. In cloud computing, the customer is provided with their requested service by the service provider and payment is done based on the service used [13]. The resource allocation to the cloud system is an important task. The author [17] uses *M/M/*1 queuing model for task scheduling. In this, the author checks the resource utility, cost, and energy to improve the performance of the cloud environment. Feng, G [18] has improved the QoS of the resource allocator based cloud environment by considering QoS parameters such as arrival rate, service rate, pricing and available resource. D. Xuet, *et al.* [27] found a solution for traffic-aware resource provisioning to improve the energy cost of the scheduling process. The QoS is optimized by enhancing the cost-efficiency of energy. In [24], the author proposed an algorithm for managing load traffic between jobs by diminishing load capacity region. It shows an optimal throughput without using the job size parameter in the algorithm. The authors [26] discuss the important issue of resource management and methods to overcome this issue occurred in Infrastructure as a Service (IaaS) cloud environment. The technique used in resource management such as provisioning, allocation, resource mapping and adaption of the resource.

P. T. Endo *et al.* [28] surveys on challenges and outcomes of resource management. The author comes out with several issues and the solution for the issues for distributed cloud system. The challenges and outcomes are represented in the step-wise modeling process. In [14], the distribution of response time is determined for the queuing model to evaluate the performance of the system and inter-arrival time and service time follows an exponential distribution. In [19], the author uses an optimal job scheduling method that is based on social allocation and cost optimization methodology. This scheduling improves the QoS of the cloud environment. H. Daochaoet *et al.*, [25] introduced dominant re-

source sharing using a virtual machine to solve the multi-server scheduling problem and to optimize the response time in distributed cloud environment. The configurations space involved in dominant resource sharing are processing memory, bandwidth and storage. Z. Xiao *et al.* [20] presented a virtualized environment for data center resource allocation that aims to minimize the number of servers involved in resource allocation. The author [15] used fuzzy logic to manage the cloud computing system. The cloud uses the user parameter, and fuzzy decision maker makes the decision when a critical situation occurs while managing the cloud. In [16], the decision making is handled by a new handler called fuzzy handler. The dynamic programming is involved in fuzzy logic to make a decision.

3. ENHANCED PRIORITY SCHEDULING ALGORITHM

3.1 System Model

The system model (as shown in Fig. 1) of the cloud environment consists of three components such as Entry point server (EPS), Service provider server (SPS) and Database Server (DS). We propose a multi-server finite system with the queuing model. This model has single entry point server (EPS). The server act as job scheduler which can prioritize jobs based on Enhanced Priority scheduling algorithm (EPSA) and forward the job to one of the service provider server SP_i where $i = 1, 2, N$. The job scheduler is represented by *M/M/*1 queue with service and arrival rate modeled on Poisson probability distribution function with parameter μ and λ respectively where $\lambda < \mu$. Its purpose is to determine the scheduling order of the jobs. In doing so, it uses the EPSA algorithm for ordering and forwarding to the service provider (SPS). The EPS uses priority scheduling. The priority is based on the attribute of jobs. The jobs attribute involved in this algorithm are execution time, resource requirement, target time, Time-to-Live, and task dependency. A service provider server (SPS) is a processor representing the physical service computing of cloud architecture. The SPS_i performs all the service to the user. Each service provider is represented by *M/M/c* queue with same service rate μ modeled on binomial pdf. The jobs that enter the SP_i will get access to the database server (DS) with a probability of δ. The required information or resource needed for execution of job will be available in DS.

3.2 Scheduling Algorithm

The proposed Enhanced Priority Scheduling Algorithm (EPSA) schedules the jobs on apriority basis. The priority of the jobs is estimated using the attribute and characteristics of jobs. The jobs that enter for the service are sent to entry point server (EPS). The EPS measures the necessary attribute required from the jobs for prioritizing scheduling. The attribute that is measured are expected execution time, resource requirement, target time of the job, CPU utilization, and dependency. The workflow of EPSA is shown in Fig. 2.

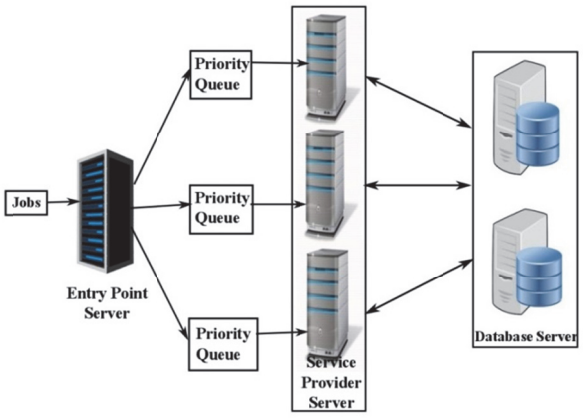


Fig. 1. Cloud-based queuing system model.

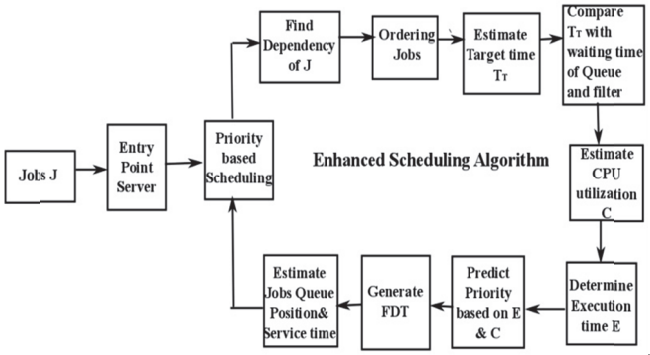


Fig. 2. Working process of Enhanced Priority Scheduling Algorithm.

3.3 Expected Execution Time

The service provider server (SPS) is *M/M/c/N* queuing system with binomial probability density function. The expected execution time is the mean time taken for execution of the jobs. Let *T* be a random variable that denotes the period for a job to reach the SP.

$E[T]$ is expected reaching time. $X \sim B(\frac{n}{E(T)}, P_N)$ defines the binomially random variable that denotes the execution time of the job in the SPS.

The Expected Execution time is

$$E[T] = \frac{n}{E[T]} P_N. \tag{1}$$

The *c* represents the number of service provider server. The *N* represents the finite number of jobs that can enter the Multi-server system. The probability P_N of *N* jobs joining the system is

$$P_N = \begin{cases} \frac{1}{N!}\rho^N P_0, N \le c \\ \frac{\rho^N}{c!c^{N-c}} P_0, N > c \end{cases}. \quad (2)$$

The $\rho=\lambda/\mu$ represents *M/M/c* server utilization. The above probability of jobs joining the system is categories into two condition bases. The Probability P_0 of the single job entering the system is given as

$$P_0 = \left[\sum_{n=0}^{tc} \frac{\rho^n}{n!} + \frac{\rho^c}{n!} \sum_{n=c+1}^{N} \frac{\rho^{n-c}}{c^{n-c}} \right]^{-1}. \quad (3)$$

3.4 Resource Requirement (R_R)

The amount of resource and time taken for accessing the resources from the database server is taken into account of the prioritization process. The probability of accessing a job in DS is δ and time taken to access the job is denoted as T_D. Let $E[T_D]$ be the expected access time. The resource requirement is

$$R_R = \delta E[T_D]. \quad (4)$$

3.5 Target Time

The target time is the period of the job to be completed. This time is very vital for prioritizing and scheduling jobs. The target time T_T creates impact in scheduling. In a cloud environment, the customer fixes target time for the task. The service provider has to submit the task within the target time. Likewise, the jobs do also have target time. Priority is given to jobs that have less target time. Target time is assessed based on the Queue's waiting time. The waiting time for a job in the queue must be greater than the target time, to allow the jobs to enter the entry point server. The waiting time fora job in the*M/M/c/N* queuing system is

$$W_T = W_q + \frac{1}{\mu}. \quad (5)$$

The waiting time of queue is defined as

$$W_q = \frac{Ln_s}{\lambda}. \quad (6)$$

The average number of customer in the system is represented as

$$Ln_s = Ln_q + \rho. \quad (7)$$

The expected length of the queue is denoted as

$$Ln_q = \frac{\rho^{c-1} p_0}{(c-1)!(C-\rho)^2} \left\{ 1 - \left(\frac{\rho}{c}\right)^{N-c+1} - (N-C+1)\left(1-\frac{\rho}{c}\right)\left(\frac{\rho}{c}\right)^{N-c} \right\}. \quad (8)$$

3.6 Dependency

The job dependency and resource dependency creates a great impact while scheduling and executing the job. At a time, the system can execute only one job. The job dependency means the execution of one job depends on the outcome of another job. The job dependency creates linked list with other jobs and spends more time for execution. Since the dependency spends more time, the jobs that are dependent are given less priority while scheduling. The resource dependency also creates a similar impact on scheduling. At a time, only one job has access to the database server (DS) for the resource. If the jobs have resource dependency, the job has to wait for some amount of time, and then only it can access. So the job that has resource dependency will have less priority since waiting time is wasted in resource dependency. The dependency of jobs is rated on the probability of a job that can depend on other jobs or resource for execution. The probability value Ω will be between 0 to 1 that determines the dependency of the jobs while prioritizing.

3.7 CPU Utilization

The CPU utilization is also considered in the prioritization attribute of jobs. The CPU utilization is the amount of time the CPU is not idle. Based on workload and jobs weight-age the CPU is allocated. The CPU utilization of job is calculated as

$$CPUUtilization, C_U = \frac{ExpectedExecutionTime}{ServerServiceTime} \times 100 = \frac{E[X]}{\mu} \times 100\%. \quad (9)$$

Table 1. Notation and description.

Notation	Description
$J_1, J_2, \ldots, J_N$	'N' jobs arriving the system
$E[T]$	Expected Execution Time
N	Finite number of jobs entering the system
P_N	Probability of n job joining the system
C	Number of service providing servers in a Multi-tier cloud
μ	Service rate of executing a job in the server
λ	Arrival rate of a job in the system
P_o	Probability of single job entering the system
R_R	Resource Requirement
δ	Probability of accessing a job in database system
$E[T_D]$	Expected Access Time
T_T	Target time
W_T, W_q	Waiting time for a job in the system and queue respectively
Ln_s, Ln_q	Average no of customer in the system and queue respectively
Ω	Dependency
C_U	CPU Utilization

3.8 Fuzzy Decision Tree

The Fuzzy Decision Tree creates a tree form based on the priority value JP of the jobs. The fuzzy decision tree sees all possibility and then only predicts the order of the jobs in the tree. The tree is drawn on ascending value of JP.

3.9 Enhanced Priority Scheduling Algorithm EPSA:

At the Entry Point server, the 'N' jobs enter and scheduling start as follows,

1. Compare ($J_1, J_2, \ldots, J_N, \Omega_1, \Omega_2, \ldots, \Omega_N$)
   (a) Order ($J_1, J_2, \ldots, J_N$) based on Ω value.
2. Estimate the target time of jobs $J_1, J_2, \ldots, J_N$
   (a) Target time ($J_1, J_2, \ldots, J_N$) = $JT_1, JT_2, \ldots, JT_N$
   (b) Calculate W using μ & λ
3. Compare ($W, JT_1, JT_2, \ldots, JT_N$)
   (a) If ($W > JT_1, JT_2, \ldots, JT_N$)
   Then, $J_1, J_2, \ldots, J_N \rightarrow$ next level.
   (b) Else
   Then, $J_1, J_2, \ldots, J_N \rightarrow$ eliminated.
4. Compare ($E[X], C_U$)
   (a) Execution time ($J_1, J_2, \ldots, J_N$) = $E[X_1], E[X_2], \ldots, E[X_N]$
   (b) CPU utilization ($J_1, J_2, \ldots, J_N$)= $C_{U1}, C_{U2}, \ldots, C_{UN}$
   (c) Priority ($JP_1, JP_2, \ldots, JP_N$) = $E[X_1] + C_{U1}, E[X_2] + C_{U2}, \ldots, E[X_N] + C_{UN}$
5. Fuzzy Decision tree, FDT ($JP_1, JP_2, \ldots, JP_N$)
   (a) The fuzzy decision tree is generated based on the $JP_1, JP_2, \ldots, JP_N$ value.
   (b) The jobs are ordered based on the tree.

After the ordering is done at the entry point, the Entry Point Server (EPS) allocates the service to the job based on the order. The jobs are allowed to enter the Service Provider Server (SPS) depending on their order level. The efficiency of the algorithm lies in it estimation process of jobs service time and level of service. It tells the job at which position it stands and at which time it will enter the SPS. This estimation differs frequently based on the server capacity. The Enhanced Priority scheduling algorithm (EPSA) predicts the position and timing of job service. It helps to improve the performance and Quality of Service (QoS) in the queuing model. The prediction is made using fuzzy logic.

4. PERFORMANCE EVALUATION

In this, the simulation of the queuing system is done using JSIMgraph Tool (JMT) [22]. The performance evaluation of the proposed system is analyzed. The queuing system is constructed with two service provider server, one entry point server, and two database servers (as shown in Fig. 3). The proposed priority based scheduling named Enhance Priority scheduling algorithm (EPSA) is implemented in the constructed queuing system. This system can be Amazon EC2, Google App server *etc.* The simulation of this type of system is analyzed to predict the QoS for the proposed system. The performance

evaluation of proposed system is measured using parameters such as response time, queuing time, jobs arrival rate and utilization of the system.

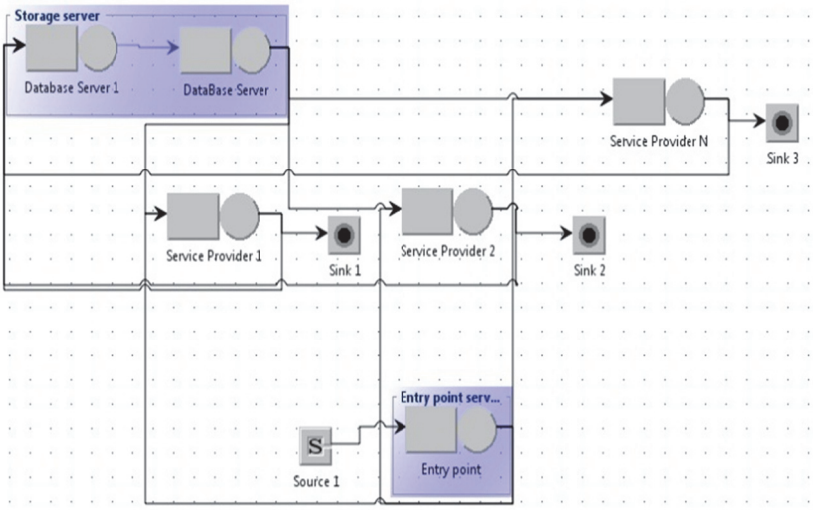


Fig. 3. Queuing Model constructed in JMT.

4.1 Utilization in Multi-Tier Queuing Cloud System

The utilization is defined as the proportion of the available that a piece of system operating their jobs. The CPU utilization is determined based on the jobs weight-age of execution and loads it require for the job service completion process. The CPU utilization is calculated for the proposed scheduling algorithm. It (Fig. 4) shows that the utilization time decrease when a number of jobs entering the Service Provider (SP). The Utilization time is decreased due to the priority based scheduling of jobs. Thus, the new algorithm reveals the best executor for priority jobs.

4.2 Response Time for Jobs

Response time is defined as a total amount of time taken to respond to request submitted. The EPSA provide less response time with more service time. It responds to all jobs that enter the service system. The EPSA (Fig. 5) shows that the response time increases as well as no of jobs increases. The proposed algorithm provide more response or service time when no of jobs increases.

4.3 Queuing Time of Jobs

The queuing time is the waiting time of the queue in the system. The ESAP (Fig. 6) shows queuing time decreases as the no og jobs increases. The waiting time allocated for a job in the queue decreases due to faster processing and utilizing less CPU time in the

server. It reveals that the waiting time for the job is very less compared to traditional algorithm.

4.4 Arrival Rate of Jobs in Queuing System

The Arrival rate is the rate at which the jobs enter the system for the service. The arrival rate decreases and reaches saturation point using proposed scheduling algorithm. The EPSA shows (Fig. 7) better performance compared to other scheduling algorithms.

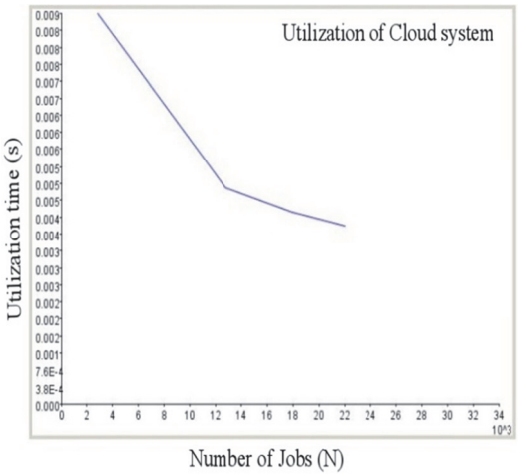


Fig. 4. Utilization cloud queuing system.

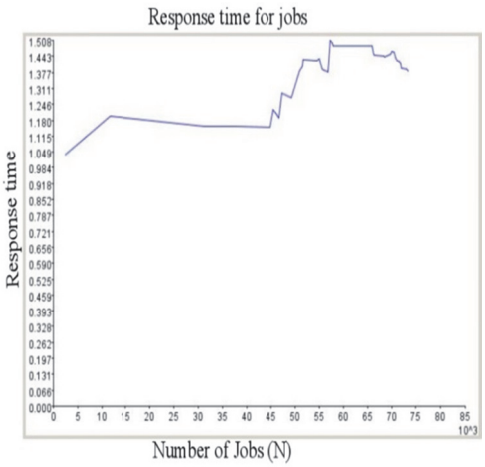


Fig. 5. Response time of Proposed queuing model.

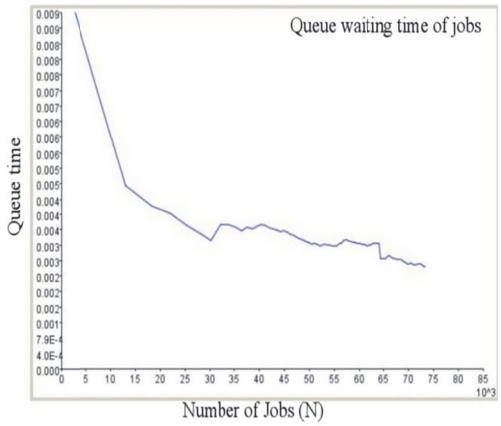


Fig. 6. Waiting time of jobs in queuing system.

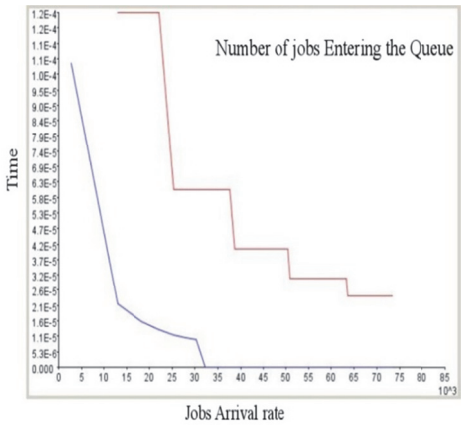
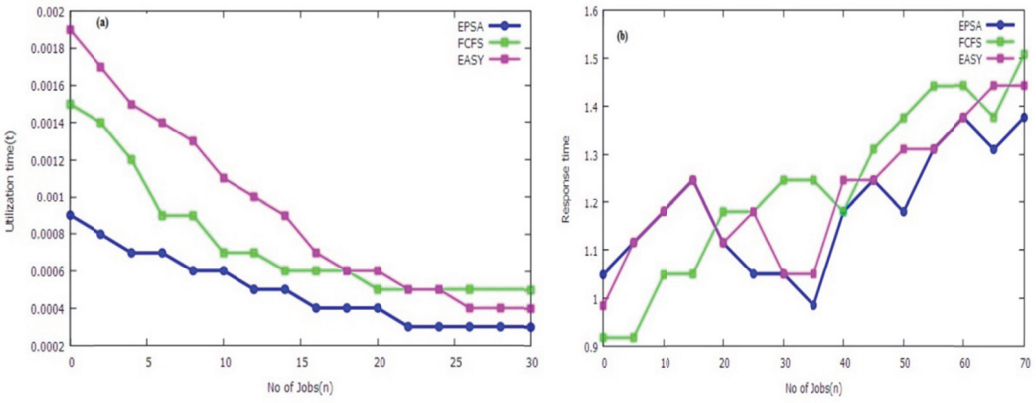


Fig. 7. Arrival rate of queuing system.

The proposed EPSA scheduling algorithm is evaluated individually using JSIM-graph tool. The proposed algorithm is compared with a traditionally proven algorithm such like First Come First Serve (FCFS) [2] and Extensible Argonne Scheduling sYstem (EASY) [3]. Many cloud systems and cloud machines follow these two algorithms being

a popular and standard algorithm in scheduling. This algorithm is compared with the proposed EPSA algorithm. In Figs. 8 (a) and (b) show that the EPSA performance efficiently than the standard FCFS and EASY algorithm.

The EPSA obtains less CPU utilization time for more number of jobs compared to FCFS and EASY. The response for the standard algorithm is more compared to the proposed algorithm. Thus the result of comparisons concludes that the proposed En- hanced Priority Scheduling Algorithm is more efficient and performance well when the number of jobs increased. The EPSA manages the resource and reduces the CPU times. This efficiency and quality improve the QoS of the Multi-Server cloud environment in the scheduling process.



(a) (b)

Fig. 8. (a) Utilization time and (b) Response time.

5. CONCLUSIONS

The proposed scheduling algorithm Enhanced priority scheduling algorithm (EPSA) is developed to improve the performance by using aset of job attributes entering the cloud queuing system. The EPSA is implemented using JSIMgraph. The result analyzed by EPSA shows enhancement in performance of the cloud queuing system. The performance is measured using utilization, response time, waiting time and arrival rate. The EPSA schedules jobs based on the priority of the jobs. The priority of the job is determined using attributes such as expected execution time, resource requirement, target time of the job, CPU utilization and dependency. The final decision of position and service time is predicted using fuzzy tree decision.

REFERENCES

1. V. Bui, W. Zhu, A. Botta, and A. Pescape, "A Markovian approach to multipath data transfer in overlay networks" *IEEE Transactions on Parallel Distributed Systems*, Vol. 21, 2010, pp. 1398-1411.

2. X. Zhu, X. Qin, and M. Qiu, "QoS-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters," *IEEE Transactions on Computers*, Vol. 60, 2011, pp. 800-812.
3. U. Schwiegelshohn and R. Yahyapour, "Analysis of first-come-first-serve parallel job scheduling," in *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998, pp. 629-638.
4. A. Mu'alem and D. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in Scheduling the IBM SP2 with backfilling," *IEEE Transactions on Parallel Distributed Systems*, Vol. 12, 2001, pp. 529-543.
5. M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informatics Journal*, Vol. 16, 2015, pp. 275-295.
6. J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," *Metaheuristics for Scheduling in Distributed Computing Environments*, Springer, 2008, pp. 173-214.
7. T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, Vol. 61, 2001, pp. 810-837.
8. U. Schwiegelshohn and R. Yahyapour, "Analysis of first-come-first-serve parallel job scheduling," in *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, Vol. 98, 1998, pp. 629-638.
9. D. A. Lifka, "The anl/IBM sp scheduling system," *Job Scheduling Strategies for Parallel Processing*, Springer Berlin Heidelberg, 1995, pp. 295-303.
10. O. Thebe, D. P. Bunde, and V. J. Leung, "Scheduling restartable jobs with short test runs," in *Proceedings of the 14th Workshop on Job Scheduling Strategies for Parallel Processing*, 2009, pp. 116-137.
11. K. Xiong and H. Perros, "Service performance and analysis in cloud computing," *IEEE World Conference on Services-I*, 2009, pp. 693-700.
12. B. Yang, F. Tan, Y. S. Dai, and S. Guo, "Performance evaluation of cloud service considering fault recovery," in *Proceedings of IEEE International Conference on Cloud Computing*, 2009, pp. 571-576.
13. J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee, and A. Y. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud," in *Proceedings of the 20th ACM International Symposium on High Performance Distributed Computing*, 2011, pp. 229-238.
14. K. Xiong and H. Perros, "Service performance and analysis in cloud computing," *IEEE World Conference on Services-I*, 2009, pp. 693-700.
15. D. M. J. dos Santos and E. A. de M. Fagotto, "Cloud computing management using fuzzy logic," *IEEE Latin America Transactions*, Vol. 13, 2015, pp. 3392-3397.
16. S. M. Chen, "A new approach to handling fuzzy decision-making problems," *IEEE Transactions on Systems*, *Man and Cybernetics*, Vol. 18, 1988, pp. 1012-1016.
17. Z. Zhang and Y. Li, "User utility-oriented queuing model for resource allocation in cloud environment," *Journal of Electrical and Computer Engineering*, 2015, p. 4.
18. G. Feng and R. Buyya, "Maximum revenue-oriented resource allocation in cloud," *International Journal of Grid and Utility Computing*, Vol. 7, 2016, pp. 12-21.

19. B. Xu, C. Zhao, E. Hu, and B. Hu, "Job scheduling algorithm based on Berger model in cloud environment," *Advances in Engineering Software*, Vol. 42, 2011, pp. 419-425.
20. Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, 2013, pp. 1107-1117.
21. R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *Concurrency and Computation: Practice and Experience*, Vol. 14, 2012, pp. 1507-1542.
22. M. Bertoli, G. Casale, and G. Serazzi, "JMT: performance engineering tools for system modeling," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 36, 2009, pp. 10-15.
23. S. T. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," *IEEE/ACM Transactions on Networking*, Vol. 22, 2014, pp. 1938-1951.
24. H. Daochao, Z. Chung, Z. Hong, and L. Xinran, "Resource intensity aware job scheduling in a distributed cloud," *China Communications*, Vol. 11, 2014, pp. 175-184.
25. S. S. Manvi and G. K. Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey," *Journal of Network and Computer Applications*, Vol. 41, 2014, pp. 424-440.
26. D. Xu, X. Liu, and A. V. Vasilakos, "Traffic-aware resource provisioning for distributed clouds," *IEEE Cloud Computing*, Vol. 2, 2015, pp. 30-39.
27. B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, Vol. 23, 2015, pp. 567-619.

G. Nalinipriya is currently working as a Professor in Department of Information Technology in Saveetha Engineering College, Chennai, India. She received her bachelor of Engineering from Madras University and Master degree in Computer Science and Engineering from Anna University Chennai. She has completed Ph.D. Degree from Anna University Chennai. She has more than 20years of experience in Teaching. She has published papers in peer reviewed journals and conferences. She is a Member of IEEE and other professional bodies such as ACEEE, ISTE, CSTA, and IAENG, IRED. Her research area includes cloud security, web security, medical data mining, big data analytics and mobile application development.

K. G. Maheswari is currently working as Assistant Professor (Senior) in Department of Master of Computer Application in Institute of Road and Transport Technology, Erode, Anna University, Chennai, India. She received her bachelor of Engineering from Madras University and Master degree in Computer Science and Engineering from Sastra University. Now she is pursuing Ph.D. in Anna University Chennai. She has more than 16 years of experience

in Teaching. She has published papers in peer reviewed journals and conferences. She is a Member of ACEEE, ISTE, CSTA, and IAENG, IRED. Her research area includes web security, cloud security, data mining and big data analytics.

K. Kotteswari is currently working as Assistant Professor in the Department of Computer Science and Engineering in Annai Mira College of Engineering and technology, Vellore, Anna University. She received her Bachelor of Technology in Information Technology from Panimalar Institute of Technology, Anna University, Chennai and Master degree in Computer Science and Engineering from Sri Venkateswara College of Engineering, Anna University Chennai. Her research interest are mobile ad hoc network, big data, cloud computing and data structure.