Received: September 26, 2017

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Parallel Queue Scheduling in Dynamic Cloud Environment Using Backfilling Algorithm

Jayapandian Natarajan[1]*

[1]*Christ University Faculty of Engineering, Bangalore, India*
* Corresponding author Email: njayapandian@gmail.com

Abstract: Cloud Computing reshapes the entire computing paradigm. In general, cloud computing means outsourcing available services and data storage in centralized scenario. In cloud computing task allocation is a major problem because multiple numbers of tasks are allocated to multiple numbers of processors for simultaneous processing. From the given list, tasks are queued according to the ascending order based on their duration. This paper is designed to solve the Task Scheduling problem, by using our proposed effective new approach of Backfilling algorithm. Depending upon the task duration, tasks are split into multiple threads for processing. Multiple thread tasks are processed in the basic concept of "gang scheduling" technique. Here we implement new backfilling algorithm concept to minimize the idle processing time of the processors. The existing Simple Backfilling Algorithm (SBA) is used to minimize the ideal time processing. Whereas comparatively Dynamic Cloud Scheduling using Backfilling Algorithm (DCBA) is designed to reduce the ideal time processing than SBA to carry out the process of both LQueue and SQueue simultaneously. At the outset, DCBA reduces the average waiting time. As mentioned the algorithm which is specified in the previous line that contains three level which represent the working speed of the algorithm. The first and second level of DCBA algorithm is comparatively similar to the performance of SBA algorithm. The maximum better performance was given in a queue size (q=1.5) by DCBA algorithm as compare to SBA algorithm. The existing type (Gang Scheduling) consist of two approaches namely Adaptive First Come First Serve (AFCFS) and Largest Job First Served (LJFS) that focus on non-parallel jobs with deadline. When compare to existing gang scheduling algorithm and SBA algorithm the average waiting time of DCBA has slight improvement in the loader level of the key. As the separation of the queue like LQ and SQ the waiting time and average waiting time is reduced comparatively.

Keywords: Backfilling, Cloud computing, Task scheduling, Gang scheduling, Dynamic cloud, Largest queue, Shortest queue, Thread, Grid computing, Load balancing.

1. Introduction

Cloud Computing is recently emerging technology into the real world combination of distributed and Grid Computing community. In cloud computing, current implementation focuses more on research problems. One of the primary problems in cloud is task allocation that is to allocate the task to perfect processor in dynamic environment in server side. Cloud computing is not a new technology; it is a new name of grid combined with virtual machine. The grid focuses on many scheduling problems which also occurrs in online cloud computing scenario. The cloud provides virtualized computing hardware in a similar to the public utility, thus it is also termed as Infrastructure-as-a Service (IaaS). So all hardware is virtualized, the cloud gives the illusion of limitless resources which can be made available to the user on-demand and can be dynamically scaled up or down, on the other hand computing refers to the applications and software platforms being offered through the cloud usually under the notation of a service model, hence called Software-as-a Service (SaaS) [1].

The task scheduling task is a sequential activity that uses a set of inputs to produce a set of outputs. Processes in limited set are statically assigned to processors, either at compile-time or at start-up. Overhead of load balancing can be avoided using some related algorithms. Grid computing techniques can be broadly categorized as centralized or decentralized, dynamic or the hybrid policies in the latest trend [2]. Hadoop system takes the centralized scheduler architecture. In static load balancing, all the information is known in advance and tasks are allocated according to the prior knowledge and will not be affected by the system. Dynamic load balancing Mechanism has to allocate tasks to the processors dynamically as they arrive. Redistribution of tasks has to take place when some processors become overloaded [3]. The cloud computing, each application of users will run on a virtual operation system, the cloud systems distributed resources among these virtual operation systems. Every application is completely different and is independent and has no link between each other whatsoever, for example, some require more CPU time to compute complex task, and some others may need more memory to store the data. Resources are sacrificed on activities performance in each individual unit of service. In order to measure direct costs of applications, every individual use of resources (like CPU cost, memory cost and I/O cost) must be measured. The direct data of each individual resources cost has been measured, in the accurate cost and profit analysis. Genetic algorithm is very dynamic and also an effective scheduling algorithm for scientific purpose. The resource policies are managed and improved in rapid manner in these algorithms [4]. Task allocation is major problem in dynamic scheduling, these genetic algorithm provide better solution for this problem [5].

The section 2 discuss about the related works that contains explanation and drawbacks of the existing system. For example the ordinal optimization uses the concept of bi-objective method for job scheduling in scientific workflow system. The main concept of this paper is being discussed in section 3. The solution for the described problem is explained. That is dynamic cloud scheduling using backfilling algorithm split the task into two queues. Such as largest Queue (LQ) and Shortest Queue (SQ) to reduce the execution time and waiting time of the task. The result and discussion of this paper is showcased in section 4 with appropriate graphs and stabilization.

2. Relates work

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [6]. In resource polling, cloud service providers computing resources are polled together in an effort to serve multiple consumers using the multi-tenancy, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand [6]. The hybrid scheduling algorithm solves the load balancing problem and reduces the overall execution time. The author use genetic algorithm and fuzzy theory concept to minimize the cost and time. In this paper the genetic algorithm of average makespan uses MACO and ACO algorithms to get appropriate values. The main drawback of this genetic algorithm is that the imbalance in average makespan value. [7]. The author proposed scheduling algorithm named as BaRRS (Balanced and file Reuse-Replication Scheduling). These algorithms split the single job into multiple sub jobs and balance the jobs by using parallelization technique. In this BaRRS proposes methodology the set of dependency pattern is necessary for inheriting the diverse. The impact feasibility of scheduling strategy is time overheated for the scheduling process [8].

The concept of bi-objective method is job scheduling in scientific workflow system by using ordinal optimization. The main advantage of this method is to reduce the overall scheduling time. The realistic cloud computing platform is to reduce the execution and scheduling time in dynamic environment [9]. In backfilling scheduling generally measured two things. First one is prediction accuracy another one is to measure the scheduling performance [10].

Cloud computing is a combination of hardware and software infrastructure motivated by real problems appearing in advanced research area. The understanding of cloud is distributed over computing that coordinates the organizational resources sharing to high end computational applications. The main aim of cloud computing is resource virtualization. The cloud system consists of two parts both gang and simple task, that requires service. Then gang enters the system as cloud tasks, while the local tasks are simply only one task, but both tasks compete for the same resources. This part the local tasks will be given higher importance. The Cloud task that enters the system will first be dispatched to a specific site by the cloud scheduler,

and then in future it is allocated to a processor by the local scheduler. The local task that enters the system arrives directly at the local scheduler. Then the cloud scheduler has its own queue where cloud tasks are stored temporarily if specific conditions are not met.

The cloud tasks enter the system as parallel which means gangs; the local tasks are simple sequential tasks that require only a single processor for execution. In general gang consists of a number of tasks that must be allocated to a different processor. Suppose backfilling is not implemented a large gang tasks are waiting for the resources to become available will block smaller and faster tasks being it that requires execution that type of causing serve fragmentation in the system. This backfilling technique allows task to begin and finish execution before the gang, so this type of technique will improve the system performance. Task scheduling is a multisite system. The concept of backfilling algorithm is to allow the shortest job in their particular time interval; it doesn't delay in the job queue. This algorithm is to monitor the running time of the jobs to control the violation. The cloud computing scheduling algorithm named as SHARP (Scheduling of jobs and Adaptive Resource Provisioning). This algorithm should process single task into multilevel processing. The main advantage of this algorithm is to handle multiple resources in dynamic environment. This SHARP performs the jobs beyond their deadline for the satisfaction of the customer which means that sometimes it violates the rules provided by the end user [11].

The combination of optimization and scheduling algorithm produce the better result. The main advantage of this proposed method is that it consumes only 30% energy has been consumed [12]. The hybrid cloud problems solve profit maximization algorithm. The combined algorithm of simulated annealing particle swarm optimization algorithm is to solve cloud scheduling problem. The main advantage of this technique is to solve private cloud and hybrid cloud task scheduling problem. This proposed methodology is in the theoretical approach and it has not been implemented practically. It doesn't implement the realistic cloud scheduling method [13]. Zhang and Zhou proposed dynamic cloud scheduling algorithm based in Bayes classifier principle and virtual machine concept. The cloud tasks are dynamically allocated in Virtual machine. But in this approach the energy consumption for the two stage strategy is comparatively maximum than the normal energy usage [14].

3. Proposed method

The existing method, task is scheduled to available free processors based on their task parameters using scheduling algorithms. The proposed method, the given task is scheduled to available processors using dynamic cloud scheduling with backfilling algorithm. It will give significant improvement of results in execution time when compared with existing results.

The proposed system, Fig. 1 the number of task is scheduled in available free resources. The number of task can be considered as M and the number of processors is considered as N. Then find the execution time for each task. Determining the threads for each task and arrange the task in descending order with two separate queues based on their execution time. The largest execution time tasks are entered into LQueue (Largest Queue) and the shortest execution time tasks are entered into SQueue(Shortest Queue). The current task from each queue can be scheduled to available processors using this Dynamic cloud scheduling Backfilling Algorithm. Before starting the procedure calculates maximum wait time for each task that is MWT.

$$MWT = \frac{\sum_{t=1}^{Ms} P\ E_t}{M_s} \quad (1)$$

In Eq. (1), Et=Processor Execution for task, Ms=Served Task.

The overall task execution time is divided by the number of served task and provides maximum waiting time of task. Consider the CS (Cloud Scheduler) to allocate the task to processors. The Cloud Scheduler considers the Local Server Scheduler (LSS) using two queue levels. For each scheduler the mean inter-arrival time of CS, LQueue and SQueue is exponentially distributed with mean of 1/λ1 and for LQueue 1/λ2, 1/λ3 for SQueue and 1/λ4 for local in site1, 1/λ5 for local in site2 and similarly 1/λ6 for local in site2 and 1/λ7 for locals in site4. Where 1/λ1, 1/λ2, 1/λ3, 1/λ4, 1/λ5, 1/λ6, 1/λ7 are arrival rates for locals in site 1,2,3,4, and CS.

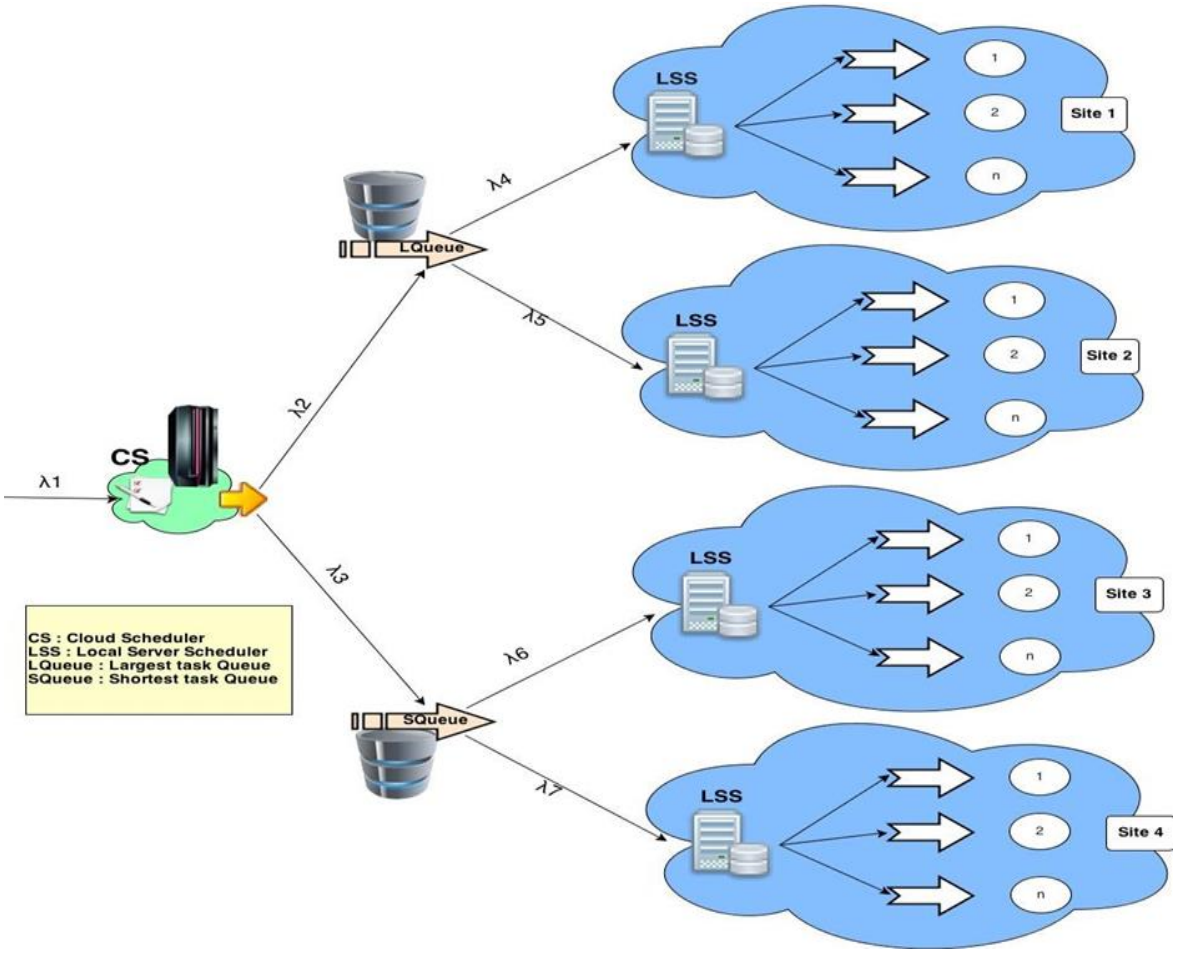

Figure.1 Dynamic Cloud Scheduling Model

We assume that arrival rates in all sites are same ($\lambda 2 = \lambda 3 = \lambda$) and that arrival rate of cloud task much be lower than that of local tasks ($\lambda 1 << \lambda$). The service time of local task or CS task is also exponentially distributed with a mean of $1/\ \mu 1$.The LQueue task are again considered by mean of arrival time $\lambda 4$ and $\lambda 5$ and then scheduled based on that parameters. We assume here that ($\lambda 4 = \lambda 5 = \lambda$) and arrival rate of $\lambda 2$ is less ($\lambda 2 << \lambda$) the service time of local task is exponentially distributed with a mean of $1/\ \mu 2$. In similar way the SQueue tasks are considered by mean of arrival time $\lambda 6$ and $\lambda 7$ and then scheduled based on this execution time. We assumed here that arrival time is same ($\lambda 6 = \lambda 7 = \lambda$) and arrival time of $\lambda 3$ is less ($\lambda 3 = \lambda$) than the service time of local task is exponentially distributed with a mean of $1/\ \mu 3$.

The communication between each site and scheduler is solely on message passing. We consider that it is contention free and therefore communication time is negligible. However, when a cloud task is dispatched to every site, extra coordinator is needed and an overhead is added to task's service time. In this study, the local tasks have priority over other and its waiting time has been minimized. The goal is to provide a quality of service for the locals that will not affect central scheduler. The technique of dynamic backfilling is also implemented to help to achieve this goal.

Proposed Algorithm: (Dynamic Cloud Scheduling using Backfilling Algorithm [DCBA])

Step 1: Form two separate queues of tasks by arranging in descending order based on their execution time. The two queues named as LQueue for Largest task and SQueue for Shortest task.

Step 2: Define maximum length of thread. Divide the tasks into threads on this basis compute duration of thread for each task.

Step 3: Start assigning the tasks in two queues parallely, according to the position of task in the queue and calculate the Average time required that is Maximum Wait Time (MWT).

Step 4: To schedule the processors, check current task in LQueue is sufficient. If Yes, schedule the tasks. If number of available processor is insufficient to schedule, move back to LQueue and

task for that requires exactly the number of free moving and bring this task to front of LQueue and schedule it. When such a task is not found, otherwise when MWT is crossed, move on to SQueue. If current task in SQueue has sufficient free processors, then schedule this task. Else if the number of free processors is insufficient to the task, move back to SQueue and look for task that require exactly the number of free processors until MWT is reached. When such a task is found, stop moving and bring this task to front of SQueue and schedule it. (If a task which fits exactly is not available, select a suitable task that comes closest to this requirement)

Step 5: For each processor, compute earliest possible start time for next job arrange the tasks for to free in ascending order based on this start times.

Step 6: Continue assigning tasks to free processors until all tasks are scheduled, if there are any tasks remaining, and then continue from step 3.

Step 7: The task with latest completion time in LQueue adds with SQueue latest completion time gives total processing time for all the jobs. Stop the process.

4. Result and discussion

The queue scheduling model is simulated with discrete event simulation models using the independent replications methods [8]. Each result presented is based on the average value from simulation experiments with different parameters of random numbers. The standard IEEE research works has been implemented using 512 tasks allocated to 16 processors such as 8192 task per cycle. In our work we assume that tasks entered in to scheduling queue in the system are always limited and keeps a standard rate. The arrival rate λ1 is static for all experiments and consider that it is equal to 0.5, which means that the mean inter-arrival time for the number of tasks m is 1/ λ1=2. However, local task arrival rate of scheduling can modify based on the number of required tasks entered into scheduler and size of the task considered for available resource size. In the simulation experiments it is to be set mean inter-arrival time for local scheduler to 1/ λ1=0.10, 0.143, 0.15 which correspond respectively to arrival rate of the tasks λ=9.4, 7, 6.3.

The term μ which means processor Execution time assume that 1/ μ=1, which implies μ=1. At the end of execution these values were chosen and studied for scheduling schemes under different load performance. And while the task has been entered, it can be scheduled based on its size in the different queues, so the queue length has been considered at different values such as q=0.25 for measuring system performance in order to find lower level of task. And assume q=1.0 for evaluate the system performance at balanced level then q=1.5 has been considered for, to study the behaviour of the system under smaller tasks as workload.

The following results represent the difference between performances and time, cost of two algorithms based on various workloads and different task size parameters. In Eq. (2-5) the performance of scheduler results regarding Processor Execution Time, Average Execution Time, and Average Maximum Waiting Time, is PEt, AET, AMWT, and ET for arrival of task λ. Finally table list the time-to-execute efficiently for all arrival rates and task size parameters.

$$WT = \frac{MWT}{PE_T} \tag{2}$$

$$ET = \frac{1}{M}\sum_{i=1}^{m} t_m \tag{3}$$

$$AET = \frac{\sum_{i=1}^{m} P(t_j)E_j}{\sum_{j=1}^{m} P(t_j)} \tag{4}$$

$$PET = \sum_{i=0}^{n} P(t_j) \tag{5}$$

The comparison of execution time given by two different scheduling algorithms for the queue size q=0.5, 1.0 and 1.5 respectively.

Table 1. Notations used in Performance Metrics

Notations	Abbreviations
Ms	Machine
PET	Processor Execution Time
AMWT	Average Maximum Waiting Time
MWT	Maximum Waiting Time
ET	Execution Time per Task
AET	Average Execution Time
AWET	Average Weighted Execution Time
AWT	Average Waiting Time
AMWT	Average Maximum Waiting Time
T j=1 to m	Task
λ	Arrival rates of Tasks
μ	Mean Processor Execution time

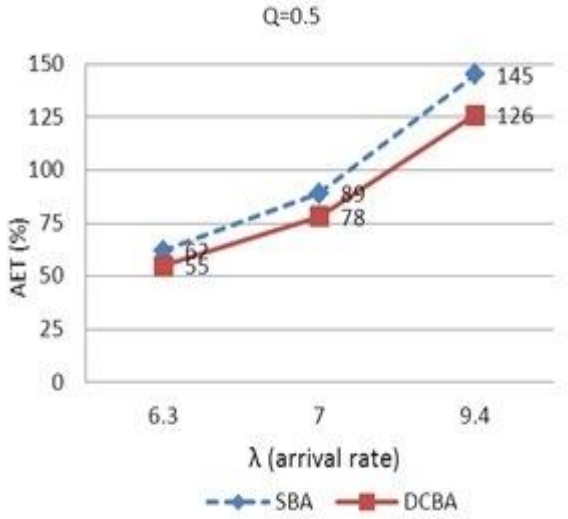

Figure. 2 AET Comparisons of SBA&DCBA for Q=0.5

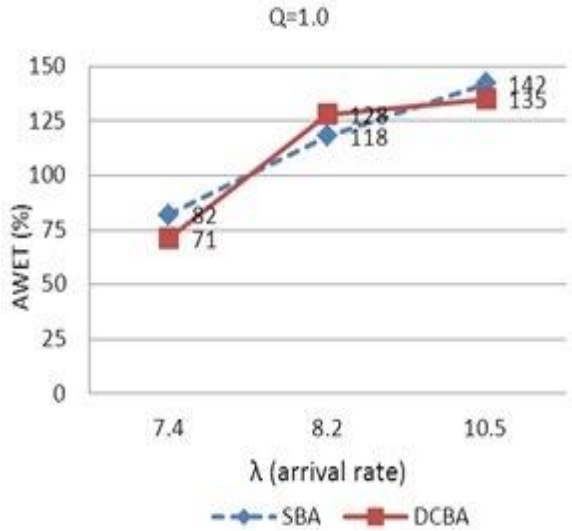

Figure. 5 AWET Comparison of SBA&DCBA for Q=1.0

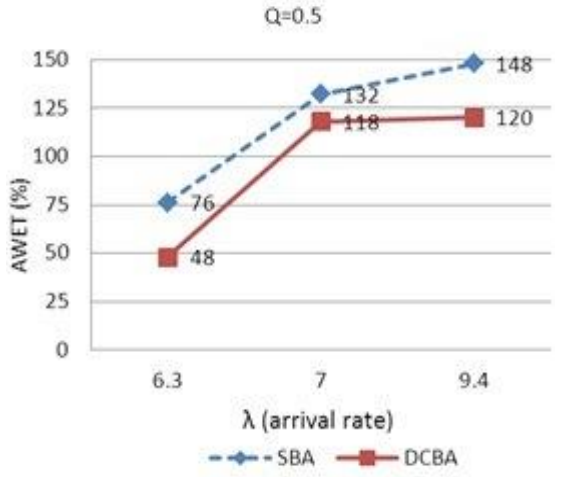

Figure. 3 AWET Comparison of SBA&DCBA for Q=0.5

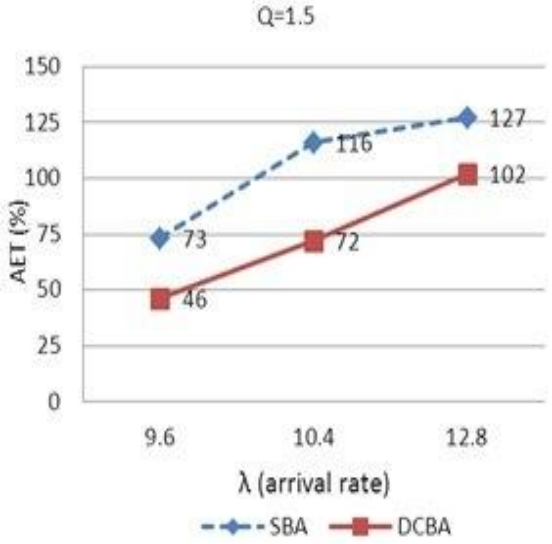

Figure. 6 AET Comparison of SBA&DCBA for Q=1.5

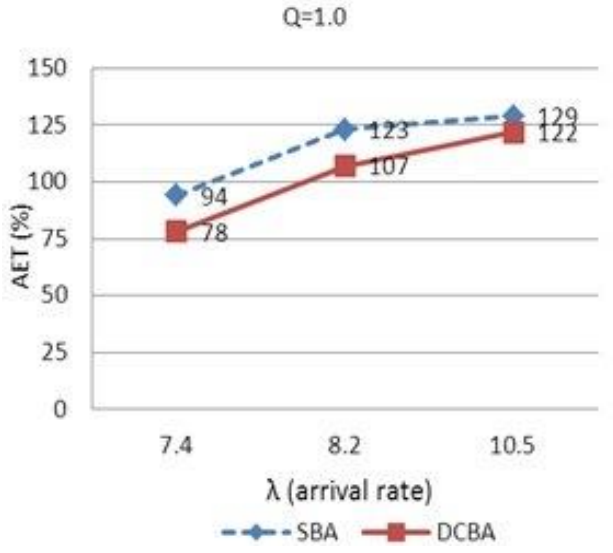

Figure. 4 AET Comparison of SBA&DCBA for Q=1.0

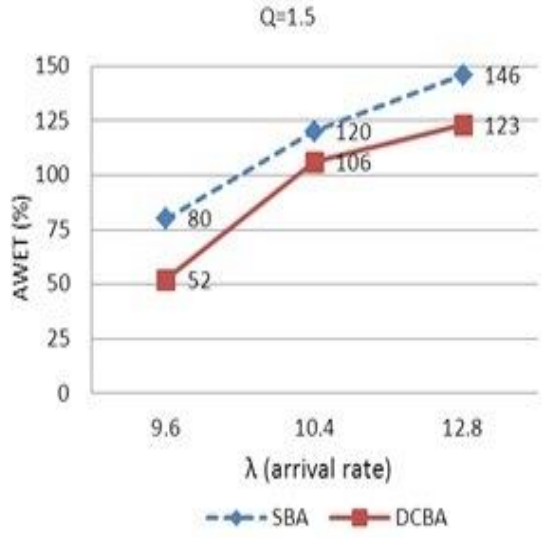

Figure. 7 AWET Comparison of SBA&DCBA for Q=1.5

Table 2. Comparison of AET vs AWET

Queue	Time	Arrival Rate	SBA (%)	DCBA (%)
Q=0.5	AET	6.3	62	55
		7	89	78
		9.4	145	126
Q=0.5	AWET	6.3	76	48
		7	132	118
		9.4	148	120
Q=1.0	AET	7.4	94	78
		8.2	123	107
		10.5	129	122
Q=1.0	AWET	7.4	82	71
		8.2	128	118
		10.5	142	135
Q=1.5	AET	9.6	73	46
		10.4	116	72
		12.8	127	102
Q=1.5	AWET	9.6	80	52
		10.4	120	106
		12.8	146	123

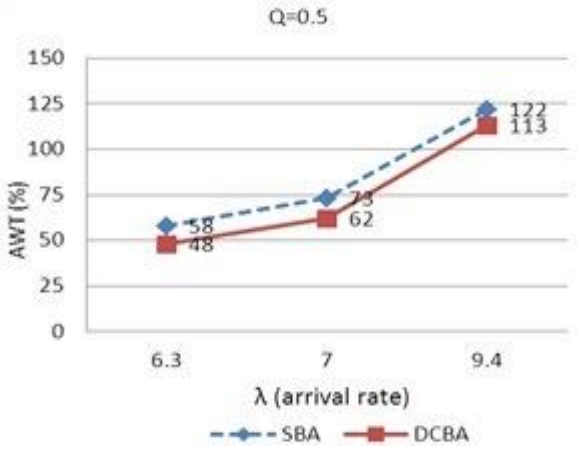

Figure. 8 AWT Comparison of SBA&DCBA for Q=0.5

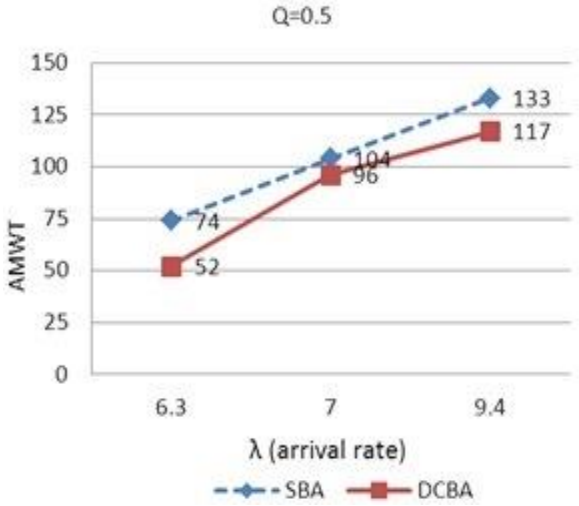

Figure. 9 AMWT Comparison of SBA&DCBA for Q=0.5

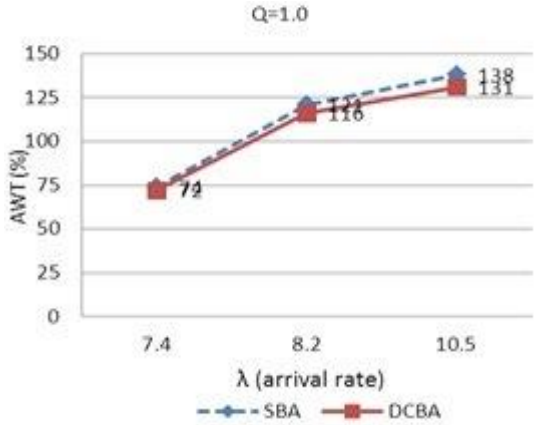

Figure. 10 AWT Comparison of SBA&DCBA for Q=1.0

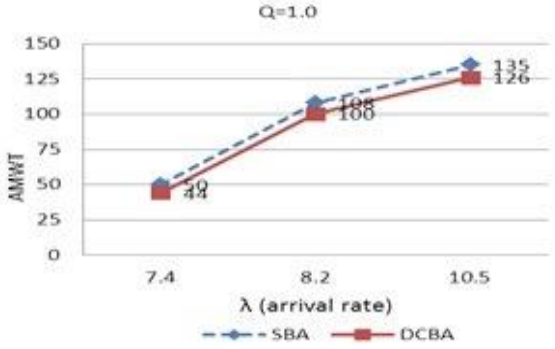

Figure. 11 AMWT Comparison of SBA&DCBA for Q=1.0

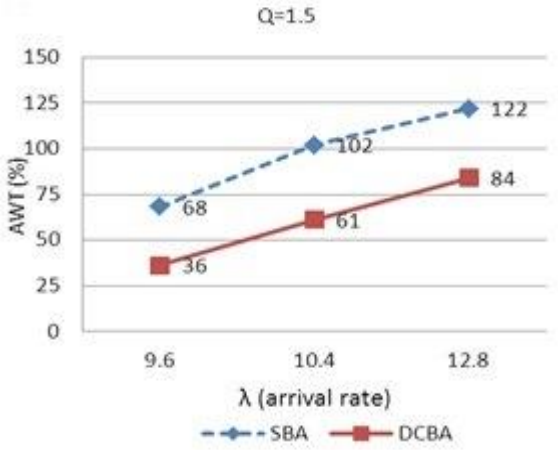

Figure. 12 AWT Comparison of SBA&DCBA for Q=1.5

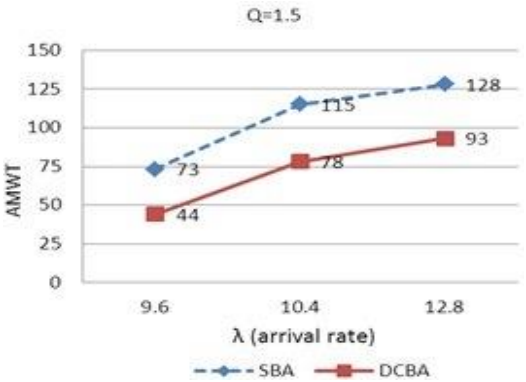

Figure. 13 AMWT Comparison of SBA&DCBA for Q=1.5

The Fig. 2, Fig. 3 and Table 2 shows the system performance comparison with new improved algorithm for the minimum queue level as q=0.5 for the minimum queue level, the task enters also less so it both these output are nearly similar. The Fig. 4 and Fig. 5 middle level as q=1.0 will slow improvement from existing method of scheduling in Fig. 6 and Fig. 7 higher level of q=1.5, when higher load are entered to the queue the average waiting time is not at highest level and which will provide better result with SBA. At minimum queue size (q=0.5), both algorithms SBA & DCBA have minimum execution time and at average queue size (q=1) these algorithm have similar execution time but DCBA has less execution time compare to SBA. At maximum queue size (q=1.5), DCBA is better performance providing algorithm compare with SBA algorithm.

The average maximum waiting time can be calculated and compared with existing method in different queue levels. Fig. 8, Fig. 9 and Table 3 is the lower load on the queue with size 0.5 as considered and compared with DCBA for slightly improvement in all levels of load on the queue. Fig. 10 and Fig. 11 queue size is 1.0 with the comparison of average waiting time comparison. The Fig. 12 and Fig. 13 queue size is 1.5 with SBA and DCBA comparison.

As it shows from the experimental results the DCBA method improved performance of the scheduling system under all situations.

Table 3. Comparison of AWT vs AMWT

Queue	Time	Arrival Rate	SBA (%)	DCBA (%)
Q=0.5	AWT	6.3	58	48
		7	73	62
		9.4	122	113
Q=0.5	AMWT	6.3	74	52
		7	104	96
		9.4	133	117
Q=1.0	AWT	7.4	74	72
		8.2	121	116
		10.5	138	131
Q=1.0	AMWT	7.4	50	44
		8.2	108	100
		10.5	135	126
Q=1.5	AWT	9.6	68	36
		10.4	102	61
		12.8	122	84
Q=1.5	AMWT	9.6	73	44
		10.4	115	78
		12.8	128	93

Table 4. Comparison of Waiting Time (min)

Arrival Rates	AFCFS (min)	LJFS (min)	SBA (min)	DCBA (min)
λ =6.3	69	56	66	50
λ=7	89	84	88.5	79
λ=9.4	131	121	127.5	115

This is because DCBA uses two separate queues and scheduling allocation each queue alternatively, so the average waiting time can be reduced. The above table 4 shows the comparison of waiting time for task scheduling; it will be done based on size of the task. The task has been entered into separate queue that is largest task entered into LQueue and smallest task entered into SQueue such that each queue is allocated alternatively for scheduling and executing the task, so that the waiting time and average waiting time can be reduced comparatively.

The waiting time for the process in backfilling algorithm is more efficient than the waiting time for same process in gang scheduling algorithm. The gang scheduling algorithm adaptive first come first serve prefers to schedule smaller job that can be easily scheduled, whereas the largest job first served prefers largest job in the scheduling cycle and performs more efficiently than adaptive first come first serve. For example the arrival rate of the process be (λ =7). The waiting time in AFCFS is 89 minutes, for the same process the waiting time in LJFS is 84 minutes [15]. The waiting time for the same process in the backfilling scheduling algorithm is much more efficient that is if the job is smaller than the waiting time of the DCBA SQ is 88.5 minutes. If the process is larger it will be handled by DCBA LQ and the waiting time is 79 minutes. Thus the proposed backfilling scheduling algorithm is much efficient and reduces the waiting time of the process as compare to the gang scheduling algorithm.

The analogy between response time and the cost for the system should be maintained in a better level, so as to make the cleared to be cost associative. The comparison between lease time and response time for the virtual machine is given by metric called cost efficiency.

$$CE = V_{LT} + V_{RT} \quad (6)$$

Where

CE - Cost Efficiency, LT - Lease Time and RT - Response Time.

Eq. (6) V_{LT} refers to the variation in LT between the stimulation experiments and V_{RT} is a variation between the response time. The negative value in CE denotes that AFCFS act better than LJFS in cost wise.

5. Conclusion

This proposed method explored the commonly used scheduling algorithm that is SBA and DCBA in cloud computing. The experimental model implemented based on existing cloud computing implementations. Multiple task sizes were considered in our implementation with dynamic and real time cloud environment. Experimental work carried out under various workloads and task size parameters. Both algorithms proved that they can be efficiently applied in a dynamic cloud environment. While both of these algorithms provide similar performance for balanced workloads and it also gives better performance when the workload gets heavier. For different task sizes the system will provide better level of performance. When the task size is large and the workload is also heavy the system provides better time performance efficiency than simple BA. This cloud study is extended in several ways. While considering the heterogeneous not only for homogeneous it's also an important for cloud environment. This method also assumed all cloud tasks in future work requiring immediate source that cloud be implemented. In future, the use of task scheduling along with various workloads and task sizes must be considered to better fit in cloud computing implementation. Thus the proposed dynamic cloud scheduling using backfilling algorithm produces comparatively less waiting time for the task that are provided by the client. Then the gang scheduling algorithm which has adaptive first come first serve and largest job first served approach for scheduling.

References

[1] S.H.H. Madni, M.S.A. Latiff, and Y. Coulibaly, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review", *Cluster Computing*, Vol.20, No.3, pp. 2489 - 2533, 2016.

[2] J. Yu and R. Buyya, "A taxonomy of scientific workflow systems for grid computing", *Journal of Grid Computing*, Vol.34, No.3, pp. 44 - 49, 2005.

[3] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors", *Journal of parallel and distributed computing*, Vol.7, No.2, pp. 279 - 301, 1989.

[4] V. Di Martino and M. Mililotti, "Scheduling in a grid computing environment using genetic algorithms", In: *Proc. of 3rd Workshop on Parallel and Distributed Scientific and Engineering Computing with Application*, Ft. Lauderdale, FL, USA, pp.235-240, 2002.

[5] Z. Zheng, R. Wang, H. Zhong, and X. Zhang, "An approach for cloud resource scheduling based on Parallel Genetic Algorithm", In: *Proc. of 3rd International Conf. On Computer Research and Development*, Shanghai, China, pp.444-447, 2011.

[6] P. Mell and T.Grance, Draft NIST working definition of cloud computing, *NIST*.2011.

[7] S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu, and A. Abraham, "Hybrid job scheduling algorithm for cloud computing environment", In: *Proc. of 3rd International Conf. On Innovations in Bio-Inspired Computing and Applications*, pp.43-52, 2014.

[8] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A.Y. Zomaya, "A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems", *Future Generation Computer Systems*, Vol.74, No.1, pp. 168-178, 2017.

[9] F. Zhang, J. Cao, K. Li, S.U. Khan, and K. Hwang, "Multi-objective scheduling of many tasks in cloud platforms", *Future Generation Computer Systems*, Vol.37, No.1, pp. 309-320, 2014.

[10] D. Tsafrir, Y. Etsion, and D.G. Feitelson, "Backfilling using system-generated predictions rather than user runtime estimates", *IEEE Transactions on Parallel and Distributed Systems*, Vol.18, No.6, pp. 789-803, 2007.

[11] D. Komarasamy and V. Muthuswamy, "ScHeduling of jobs and Adaptive Resource Provisioning (SHARP) approach in cloud computing", *Cluster Computing*, pp. 01-14, 2017, https://doi.org/10.1007/s10586-017-0976-3

[12] N. Jayapandian, A. M. J. Md. Zubair Rahman, and J. Gayathri, "The Online Control Framework on Computational Optimization of Resource Provisioning in Cloud Environment", *Indian Journal of Science and Technology*, Vol.8, No.23, pp. 1-5, 2015.

[13] H. Yuan, J. Bi, W. Tan, and B.H. Li, "Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds", *IEEE Transactions on Automation Science and Engineering*, Vol.14, No.1, pp. 337-348, 2017.

[14] P. Zhang and M. Zhou, "Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy", *IEEE Transactions on Automation Science and Engineering*, Vol.PP, No.99, pp. 1-12, 2017.

[15] I.A. Moschakis and H.D. Karatza, "Performance and cost evaluation of Gang Scheduling in a Cloud Computing system with job migrations and starvation handling", In: *Proc. of IEEE International Symp. On Computers and Communications*, Kerkyra, Greece, pp. 418-423, 2011.