International Journal of Intelligent Computing and Information Science

A SCHEDULING ALGORITHM TO ENHANCE THE PERFORMANCE AND THE COST OF CLOUD SERVICES

N. A.AL-Sammarraie
Faculty of Computers and Information
Mansoura University
Mansoura Egypt
naseemiq70@gmail.com

M. .F. Al-Rahmawy
Faculty of Computers and Information
Mansoura University
Mansoura Egypt
mrahmawy@mans.edu.eg

M. Z.Rashad
Faculty of Computers and Information
Mansoura University
Mansoura Egypt
magdi_12003@yahoo.com

Abstract.*Cloud computing is based on the pay-per-use; hence, the price of usage is one of the main factors for cloud services' customers when selecting the cloud provider to rent the service from. Hence, cloud providers need to provide competitive costs of the services for the users. Therefore, the cloud providers, in addition to optimize the utilization of the resources, aim to provide the service with the competitive cost at the same time. In order to achieve this, there is a need for a new set of economical task scheduling algorithms for the cloud. This paper introduces an algorithm for task scheduling based on assigning priorities for tasks according to their profits, where we provided examples of usage of the algorithm and compared it to some of the traditional cloud scheduling algorithms.*

Keywords: Cloud Computing, Scheduling, Priority, Resource Utilization.

1. Introduction

Cloud computing is one the up and coming most recent innovation which is growing profoundly.. There are many and different definitions of cloud computing, one of the most popular definitions was provided by NIST which defines Cloud Computing as follows [1]: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.Job scheduling is a basic activity in most computing environments; it is applied to achieve one or more necessary objectives by following a certain scheduling policy. In cloud computing, Job scheduling is one of the main approaches to increase the efficiency of the cloud environments by reducing the makespan and increase the resource utilization [3, 7]. And can be used as well to optimize the energy usage in the cloud [4, 5].

However, as cloud computing is based on the pay-per-use of the resources, one of the important issues for cloud providers companies is to provide best services with competitive cost for cloud users. Hence, there is a need for some new hybrid job scheduling algorithms for the cloud environments that aim to, in addition to optimizing resource utilization and minimizing makespan, is to provide economical and cost competitive services.In this paper we propose a new hybrid economical algorithm of scheduling in cloud computing environment. We designed this algorithm to consider two issues of cloud computing; service performance and service cost. The paper is organized as follows: in section 2, an overview of the related work is discussed. Then, in section 3, our proposed scheduling algorithm is presented. In order to evaluate the proposed algorithm, in section 4, some examples of applying it are presented and

their results are discussed. Finally, in section 5, the conclusions on the presented work are drawn and some recommended points for future work are presented.

2. Related Work

Intensive research has been conducted in cloud computing task scheduling, to solve the problem of mapping a set of tasks to a set of machines. Various algorithms have been designed to schedule the jobs in cloud computing, e.g. [8], [9], [10], [11], [12], [13], [15]. Here, we make a quick overview of two of the most commonly known algorithms, which are Min-Min [2,7], Max-Min [2,7],in addition to the ABC algorithm [6,12] which is a cost-based scheduling algorithm for the cloud.

2.1 Min-Min Algorithm

The scheduling criterion in Min-Min is to achieve Minimum Completion Time. The scheduling process is done by adding all tasks to a set known as the meta task, if the meta task not empty, the algorithm begins to calculate the completion time for each task; then, the task that has the earliest minimum execution time is taken from the set and assigned to the corresponding resource. Then, this task is removed from the meta-task set. This process repeats after removing this task till all tasks in meta-task are processed [2, 7,10].

2.2 Max-Min Algorithm

This algorithm works in a way unlike (Min-Min)algorithm method, where it choose the task which has the maximum execution time and assign it to the resource has the minimum completion time [7]. Max-Min is better than Min-Min algorithm in resource utilization cause where it assign task which has maximum execution time to the resource has minimum completion time still the tasks which has minimum completion time and assign it to another resource [2].

2.3 An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing (ABC Algorithm)

This algorithm measures the cost of the resource and applies the concept of cost-based priority by calculating the cost of each individual use of the resources and the corresponding profit of using these resources. According to these calculations, tasks are given priorities and sorted in three levels; High, Medium and Low level priority, where the tasks with highest profit are assigned the highest priority. If new task arrives its priority calculated and it is assigned to the end of the appropriate level [6].

3. The Proposed Algorithm (PCA)

Most of the traditional algorithms of scheduling in cloud computing don't make any consideration for the task's cost, where the task is assigned to any available resource as soon as it arrives. This leads to some problems such as over-costed and/or over-priced cloud services in case of high-volume simple tasks and under-costed and/or under-priced in low-volume complex ones [6]. To overcome these problems and since many people think of current cloud computing offerings as purely "pay by the drink" compute platforms [15], we proposed the ***P**erformance and **C**ost **A**lgorithm(PCA)* as a hybrid algorithm that aims not only to the minimization of the services cost paid by the user and/or maximizing the profit gained by the provider of services renting, but also aims to optimize the performance of these services by minimizing the services completion time and maximizing the resource utilization of the resources, in order to enable the provider to provide the best and most efficient services with highly

competitive prices.The base structure of the scheduler we proposed in our algorithm is composed of a number of queues equal to the number of *priority* levels considered in the system; e.g. in the examples presented in this paper, we assumed a scheduler of three different priority levels; High, Medium and Low; hence, the proposed scheduler has three different queues. Where, as explained later, the task's *priority* used by the algorithm is not the one assigned by the provider but is one calculated by the algorithm once it arrives to the scheduler according the task's cost and the profit gained from running it.

Once the task's priority is calculated, the task is sent to the appropriate queue in the scheduler, where the algorithm, shown in figure 1, assigns the task(s) in highest queues, i.e. which has/have the highest calculated priority, to the resource(s) which has the minimum completion time with a consideration of the waiting time of the tasks in the lower queues as explained next in details:

Step 1- FOR all available tasks **DO**
Calculate the *priority* of each task
END FOR

Step 2- Sort the tasks according to thepriorities in the scheduler's queues.

Step 3- FOR all tasks T_i in meta-task**DO**
FOR all resources R_j**DO**
Calculate the competition time:
$$CT_{ij} = EC_{ij} + r_j$$
END FOR
END FOR

Step 4- Find task T_k which has the highest *Priority* and assign this task T_k to the resource which has the minimum completion time.

Step 5- Remove task T_k from Meta-tasks set and update r_j for the selected R_j and Update $CT_{ij}$for all j.

Step 6-IF the waiting time of any task in the lower queues has exceeded the threshold **THEN**
Move this/these tasks to the next upper queue.
END IF.

Step 7- IF there is a new task has arrived **THEN**
Calculate its priority and sort it in the end of appropriate queue and repeat the above steps
END IF

Figure. 1. Pseudo Code of The PCA Algorithm

In Step one, we calculate the *priority* by first calculating the cost of each task on each available resource by using equation 1:

$$\text{Cost}\,(T_{i_{R_j}}) = \text{NOI}(T_i) \times \text{CPI}(T_{i_{R_j}}) + \text{D}(T_i) \times \text{CPBW}(T_{i_{R_j}}) \qquad (1)$$

Where:

NOI (T_i): is the number of instructions for task (T_i).

$CPI(T_{i_{R_j}})$:is the cost per instruction for $T_i$on resource R_j

$D(T_i)$: is the data for task (T_i).

$CPBW(T_{i_{R_j}})$: is the cost per bandwidth for running the task T_i on resource R_j.

Then, the profit is calculated for each task on the resource which has the highest cost using equation 2;

$$Profit = CostU(T_i) - CO\left(T_{i_{R_{max}}}\right) \quad (2)$$

Where:

$CostU(T_i)$: is the cost paid by the user to run task T_i

$CO(T_{i_{R_{max}}})$:is the actual cost of running task T_i on the resource R_{max}(R_{max} is the resource which has highest cost).

After that, all the tasks are sorted and each task is assigned to the appropriate queue. In our work, we proposed only three levels of queues (High, Medium and Low) and we proposed that these queues have equal range of priorities, i.e. the total range of priorities of the recent tasks is divided among the available queues. Hence, as we have three queues, we can use equation 3 to calculate the range QR of priorities for each queue:

$$QR = Profit_{max}(T_i)/3 \quad (3)$$

Where:

$Profit_{max}(T_i)$: is the maximum profit of running task T_i, it is divided by 3 as we assumed the existence of three queues in our algorithm.

Figure 2 shows an instance of our proposed schedule with three queues. The High queue has the highest region of priorities, the Middle queue has the medium region of priorities and the Low queue has the lowest region of priorities. As explained above, we can see that:

IF ($Profit(T_i) \leq QR$), THEN the task Ti is inserted in the Low queue.

IF ($QR < Profit(T_i) \leq 2QR$) THEN the task Ti is inserted in the Medium queue.

IF ($2QR < Profit(T_i) \leq 3QR$) THEN the task Ti is inserted in the High queue.

Then, in Step three, the completion time is calculated for each task on each resource in the system by using equation 4:

$$CT_{ij} = EC_{ij} + r_j \quad (4)$$

Where:

EC_{ij}: is the execution time of task T_i on the resource R_j.

r_j: is the ready time for resource R_j.

In step four, the task which has the highest priority is selected and assigned to the resource that executes it in the minimum completion time.

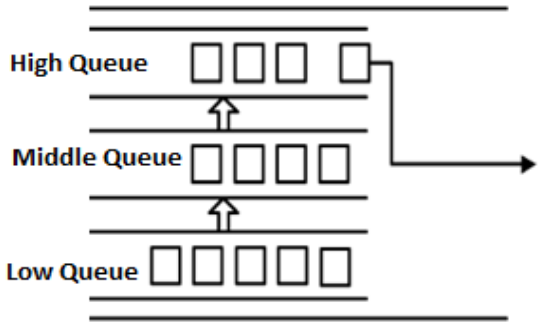

Figure.2. Level of priority

Fig. 2. Shows an instance of our proposed schedule with three queues. The High queue has the highest region of priorities, the Middle queue has the medium region of priorities and the Low queue has the lowest region of prioritiesIn step five, the assigned task is removed from the meta-task and already times and completion times for all resources are update.

In step six, in order to overcome the problem of infinite waiting of tasks in the lower queues, we assumed an aging threshold value for the maximum waiting time of a task in a queue, for all queues other than the High queue; if this task outstrips the threshold, the task migrates to the end of next upper queue.

In step seven, if a new task arrived, its priority is calculated using equation (1) and the process is repeated from step one, where all the calculations are remade and the queues are updated.

4. Results

In order to validate our algorithm, we developed a simple implementation of the algorithm using the Java language. Here, we present some examples that illustrate its work, where we used a set of metrics as a performance metrics in order to evaluate the performance of the algorithm and to compare it with some of the traditional cloud scheduling algorithms. Next, we present the performance metrics used for the evaluation; then, we present the examples.

4.1 Performance Metrics

Depending on what scheduling performance is desired in the cloud, there exist different performance metrics for evaluating different scheduling algorithms. Here, the results are evaluated on the basis of the following performance metrics.

- **Priority:** it is calculated for each task in the meta-task set based on the cost and profit of the service task and its maximum value is used to define the boundaries of the scheduler queues as stated previously.

- **Makespan:** it is the time difference between the start and finish of the sequence of jobs or tasks t_i. It can be calculated using the equation

$$\text{makespan} = \max(\text{CT}_i)_{t_i \in \text{MetaTask}} \quad (5)$$

In general, the lower the makespan, the better is the scheduling.

- **Average resource utilization rate:** it is calculated according to equation 6 borrowed from (7):

$$\text{ru} = \frac{\sum_{j=1}^{m} \text{ru}_j}{m} \quad (6)$$

Here, ru_j is the average resource utilization rate of resource j. It can be calculated using equation7.

$$\text{ru}_j = \frac{\sum(\text{te}_i - \text{ts}_i)}{T} \quad (7)$$

Where, $\boldsymbol{te_i}$and$\boldsymbol{ts_i}$are the end time and the start time of executing the task $\boldsymbol{t_i}$ on the resource $\boldsymbol{m_j}$ respectively, and **T** is the total application time so far, it can be calculated using following equation

$$T = \max(te_i) - \min(ts_i) \quad (8)$$

- **Provider Cost**: It is the cost afforded by the provider to present the service to the user. It can be calculated using equation 9:

$$\text{CO}_{\text{provider}} = \sum_{i=1}^{n} [\text{PCost}(R(T_i)) \times \text{Size}(T_i) + \text{BWCost}(R(T_i)) \times \text{Data}(T_i)] \quad (9)$$

Where:

n: is the number of scheduled task.

$R(T_i)$: is the resource chosen to run task T_i

$PCost\ (R(T_i))$: is the cost of executing task T_i on the resource$R(T_i)$.

$Size(T_i)$: is the Size of instructions executed by T_i.

$BWCost(R(T_i))$: is the bandwidth cost of running the data of task T_i on the resource$R(T_i)$.

$Data(T_i)$: is the size of Data transferred by task T_i to/from the resource.

4.2 Example 1:

The aim of this example is to illustrate the basic functionality of the proposed algorithm. In this example, it is assumed that there is a cloud environment with three resources R_1, R_2, R_3. The processing speed of these resources and the bandwidth of their communication links are shown in Table 1.

Also, assume we have a meta-task of twelve tasks T1, T2..., T12are in the meta-task, and the cloud manager is supposed to schedule all the tasks within this meta-task on the three available resources R1, R2 and R3. Table 2 represents the size details of both the instructions and data for all the tasks T1 to T12.

Table 1. Specification of the Resources

Resources	Processing speed (MIPS)	Related Bandwidth (Mbps)
R1	50	100
R2	250	200
R3	100	150

Table 2. Specification of Tasks

Task ID	instructions (MI)	Data (Mb)
T1	215	75
T2	320	95
T3	183	52
T4	198	201
T5	324	102
T6	55	63
T7	45	33
T8	600	450
T9	99	29
T10	508	307
T11	222	66
T12	403	142

Table 3 describes the actual costs of the three resources; including the processor cost in instructions per second (IPS) and the bandwidth cost in bandwidth per second (bps).

From the above specifications, we can calculate, as shown in the three columns of Table 4, the cost of running each task on each of the three resources (R_1, R_2 and R_3).

Table 3. Costs of Using the Resources			
Cost/Resource	R1	R2	R3
Cost of processor (IPS)	0.02	0.05	0.03
Cost of bandwidth (bps)	0.01	0.03	0.02

Table 4. Data of Example 1

Task ID	Actual Cost CO($) of running on			Cost CostU($) Paid the User	Profit for R_{max}
	R1 (CR1)	R2 (CR2)	R3 (CR3)		
T1	5.05	13	7.95	15	2
T2	7.35	18.85	11.5	25	6.15
T3	4.18	10.17	6.53	16	5.83
T4	5.97	15.93	9.96	20	4.07
T5	7.5	19.77	11.75	30	10.23
T6	1.73	4.04	2.91	10	5.96
T7	1.23	8	2.01	12	4
T8	16.5	43.5	27	60	16.5
T9	5.1	13.08	7.98	17	3.92
T10	13.23	34.61	21.38	50	15.39
T11	2.27	5.82	3.55	10	4.18
T12	9.48	24.41	14.93	40	15.59

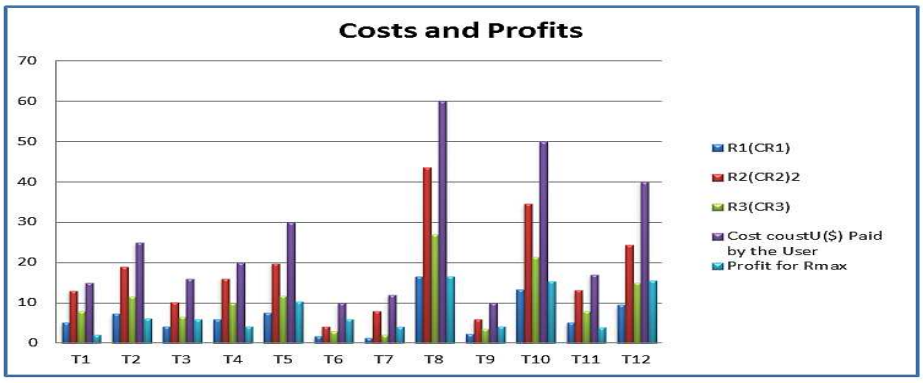

Figure.3.A Gantt chart of The Data in Table 4

In the 5th column of the same table, Table 4, the price offered to the user for running each task in the cloud environment is shown. Hence, as required for equation 2 of the algorithm, in the last column the

approximate value of the lowest profit that can be gained from running each task can be deduced by subtracting the price payable by the user, i.e. the value in the 5th column, from the maximum possible actual price, i.e. the maximum value in columns 2, 3, 4. For instance, Task 2's lowest profit = 25-maximum (7.35, 18.85, 11.5) = 6.15 as shown in the last column. Figure 3shows a Gantt chart that summarizes the data shown in Table 4.

We can deduce from the last column of table 4 the value of $Profit_{max}(T_i)$to be 16. Then, according to equation 3 the range of priorities for each of the three queues QR=16/3=5.19.Hence the limits of the three queues are as follows:

- Low queue: tasks with priority [0, 5],
- Medium queue: tasks with priority [5, 10],
- High queue: tasks with priority above 10,

This results in the distribution shown in Table 5.
According to the algorithm, the next step is to find task T_i which has the highest *priority* and assign task T_i to the resource which has the minimum completion time.

Table 5. Assigned Priority Queues for each task

Task ID	Priority Queue
T1	LOW
T2	MEDIUM
T3	MEDIUM
T4	LOW
T5	MEDIUM
T6	MEDIUM
T7	LOW
T8	HIGH
T9	LOW
T10	HIGH
T11	LOW
T12	HIGH

Table 6. Assigned Resources and the Completion Time

Task ID	Assigned Resources	Completion Time
T8	R2	4.65
T10	R3	7.06
T12	R2	6.97
T5	R1	7.5
T2	R2	8.72
T6	R3	7.96
T3	R2	9.72
T11	R3	10.62
T4	R2	11.51
T7	R1	8.73
T9	R1	11
T1	R2	12.75

Table 6 shows the resource assigned by our proposed algorithm for running each task and the *Completion Time* at which this task finishes its execution at this resource. According to equation 5, the *makespan* of this meta-task equals 12.75 which is the maximum completion time of all the tasks.

Table 7. Provider Cost	
Algorithms	Provider Cost
Min-Min	174.58
Max-Min	199.91
ABC	162.35
PCA	162.59

Also, table 7 shows the costs afforded by the provider as calculated from equation 9 for our algorithm and the three other algorithms (Min-Min, Max-Min and ABC).

As seen from the table our algorithms achieved the 2nd minimum cost after the ABC algorithm with a cost very close to it. But when looking to both the makespan, shown in figure 4, and resource utilization, shown in figure 5, we find that our algorithm beats clearly not just the ABC algorithm, but the Min-Min and Max-Min as well. This means that our algorithm can enable the cloud provider to present relatively high performance services to the users with economic and competitive prices.

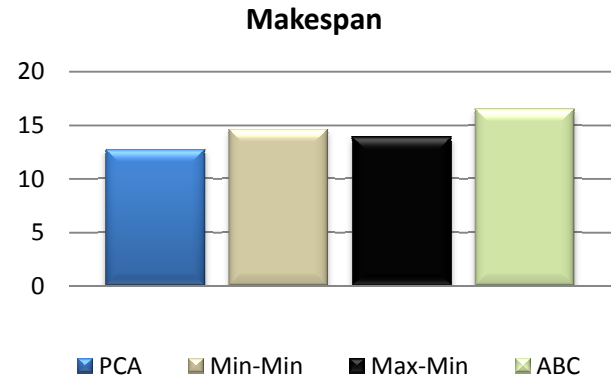

Figure. 4. Makespan Comparisons

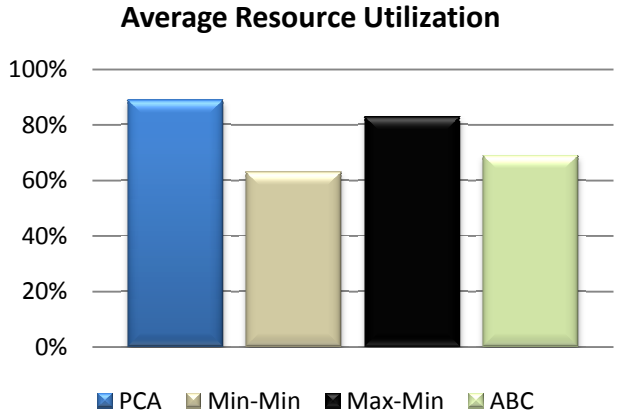

Figure. 5. Comparison of Average Resource Utilization

4.3 Example 2

The aim of this example is to show how the algorithm handles the low priority tasks to avoid starvation. In this example we assumed that we have 12 tasks, see their specification in table 8, arrived at different times (time 0, time 5 and time 10) and there are only two resources with the specification shown in table 9.

Table 8. Specification of the Tasks

Task ID	instructions (MI)	Data (Mb)	Arrival Time (Time Unit)
T1	215	75	0
T2	320	95	0
T3	183	52	0
T4	198	201	0
T5	99	29	5
T6	508	307	5
T7	222	66	5
T8	403	142	5
T9	324	102	10
T10	55	63	10
T11	45	33	10
T12	600	450	10

Table 9. Specification of the Resources

Resources	Processing speed (MIPS)	Related Bandwidth (Mbps)
R1	50	100
R2	100	150

Table 10 describes the costs of the two resources; including the processor cost in instructions per second (IPS) and the bandwidth cost in bandwidth per second (bps).

Table 10. Costs of Using the Resources

Cost/Resource	R1	R2
Cost of processor (IPS)	0.02	0.05
Cost of bandwidth (bps)	0.01	0.03

To simplify the calculations, table 11 shows approximate values of the calculated Profit for R_{max} for each of the 12 tasks as described in the earlier examples.

Table 11. Calculations of the Profit for R_{max}				
Task ID	Actual Cost CO($) of Running on		Cost CostU($) Paid by the User	Approximate Profit for R_{max}
	R1 (CR1)	R2 (CR2)		
T1	2.27	5.82	13	7
T2	13.23	34.61	50	15
T3	5.1	13.08	17	4
T4	9.48	24.41	40	15
T5	5.05	13	15	2
T6	7.35	18.85	25	6
T7	4.18	10.17	16	6
T8	5.97	15.93	20	3
T9	7.5	19.26	30	11
T10	1.73	4.04	10	6
T11	1.23	8	12	4
T12	16.5	43.5	60	16

In this example, we assumed the threshold of waiting time for each task in any queues other than the High queue is 10 timeunits.After this time, each task migrates to the next upper queue.

Table 12. Assigned Resources and the Completion Time of the Tasks		
Task ID	Assigned Resources	Completion Time
T2	R2	4
T4	R1	6
T1	R2	7
T6	R2	14
T7	R1	11
T12	R2	24
T9	R1	18
T3	R1	22
T10	R1	24
T8	R2	29
T5	R1	26
T11	R1	27

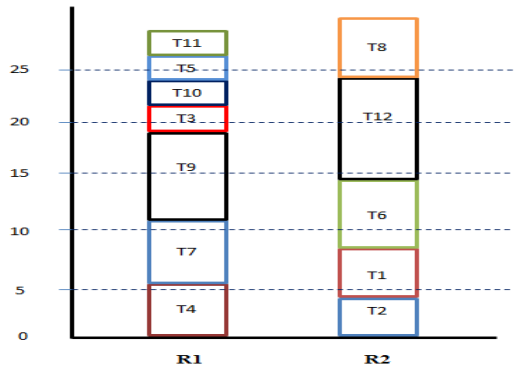

Figure.**6.**Execution of the Tasks of Example 3 on R1 and R2

By inspecting the tasks in table 8, we notice that only 4 tasks arrives at time 0; hence, from table 11, we can see that the maximum profit from these tasks equals 15 approximately; hence QR≈5 at the beginning of execution of the scheduler. Therefore, tasks T_2 and T_4 are inserted in the High queue, while task T_3 is inserted in the Low queue and task T_1 is inserted in the Medium queue. So, as shown in table 12 and in figure 8, tasks T_4 and T_2 are assigned to R_1, R_2 directly, while T_1 and T_3 waits in their queues. Once, T_2 finishes its execution and R_2 becomes free, the algorithm assigns T_1 to R_2 at time 4. At time 5, in addition to the remaining task T_3 in the Low queue, new tasks (T_5, T_6, T_7, T_8) arrive; hence, according to our algorithma new value is calculated for QR using the profit value of the remaining tasks and the newly arrived tasks to, where in this case as the calculated value of QR equals 6/3=2 which is less than the old value of QR, the algorithm keeps the old value of QR and ignores the new value. So, tasks T_6 and T_7 are inserted in the Medium queue, T_8 are inserted at the end of the Low queue after T_3, which remains in the same queue as the aging threshold has not reached yet. Later, at time 10, when tasks T_9, T_{10}, T_{11}, T_{12}, the algorithm finds that tasks T_3, T_5, T_8 has not executed yet, but only task T_3 has waited 10 time units which is the threshold value of the waiting time in any queue in our algorithm. So, the algorithm keeps the priorities of both T_5 and T_8, while in order to move task T_3 to the upper queue, the algorithm updates the value of its priority to a value of 6 which is the minimum value of the priority in the next upper queue. Then, again the algorithm recalculates a new value of QR, which becomes in this case 16/3≈5 which is similar to the old value, so the limits of the queues are kept the same. Hence, tasks T_9, T_{12} are inserted in the High queue, T_{10} is inserted in the medium queue and T_{11} is inserted in the Low queue. Later, at time 15, the algorithm finds that T_5 and T_8 have spent 10 time units in the Low queue, so it changes their priority to a value of 6 and move them to the Middle queue. Later at time 20, the algorithm finds that T_{10} has reached the waiting time threshold, so it updates its priority to a value of 11 in order to move it to the High queue. So, as seen in figure 4, at time 22 T_{10} is executed on R_2; then, at time 24, T_5 is assigned to R_2. Finally, at time 24, task T_8 is assigned to R_2. At time 25, the value of T_{11} does not change. Finally, at the time 26, T_{11} is assigned to R_1. The completion time of all tasks are shown in table 12 and figure 6, where we can see that the makespan in this example=29.

4.4 Evaluation with Other Algorithms

To evaluate and compare our proposed scheduling algorithm with three well-known cloud scheduling algorithms (Min-Min, Max-Min and ABC), we developed a simple implementation of the algorithm using the Java language, where we assumed 8 resources and 50 tasks. The following three scenarios are taken to perform the experimental testing:

1. **Scenario 1:** - Many high priority tasks along with few medium and low tasks.
2. **Scenario 2:** - Many medium priority tasks along with few high and low tasks.
3. **Scenario 3:** - Many low priority tasks along with few high and medium tasks.

The makespan for the four algorithms in each of the above three scenarios are shown next in figure 9, where it can be seen that the proposed algorithm is more efficient than the other three algorithms (Min-Min, Max-Min, and ABC) as in all the scenarios it achieves better makespan than the other algorithms.

Also, figure 10 shows the average resource utilization for the four algorithms, where it is clear that the proposed algorithm achieves the best resource utilization when compared with the other three algorithms (Min-Min, Max-Min and ABC).

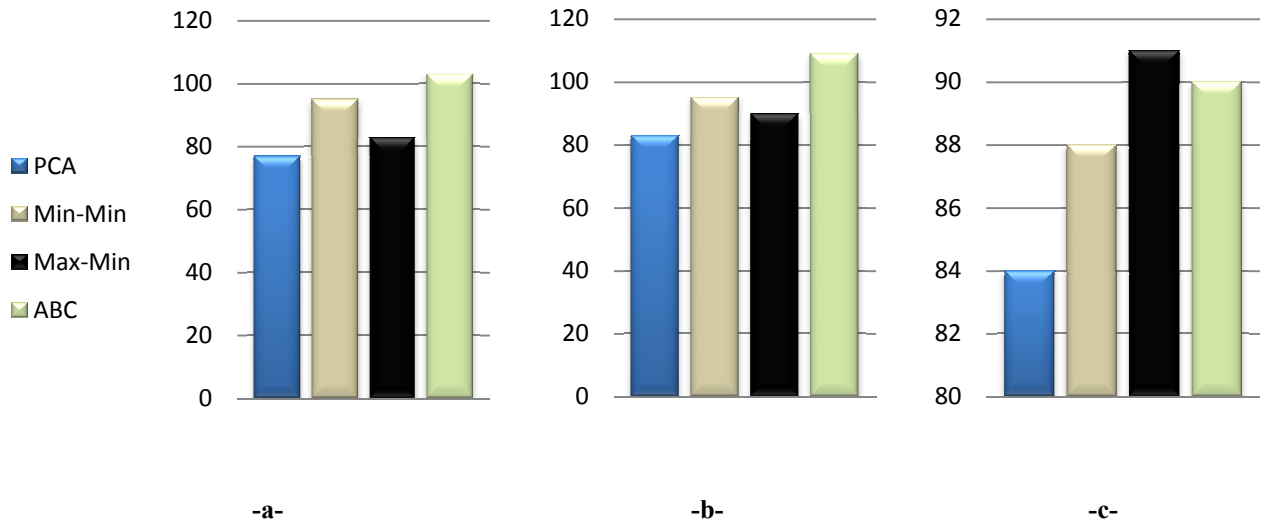

Figure.7. The Makespan for (a) Scenario 1 (b) Scenario 2 (c) Scenario 3

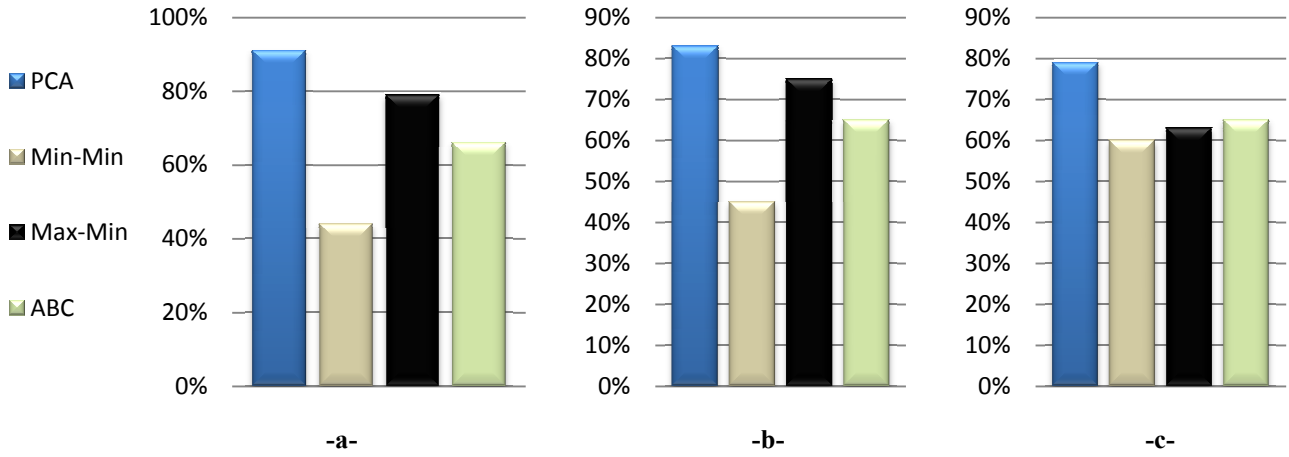

Figure.8. The Average Resource Utilization for (a) Scenario 1 (b) Scenario 2 (c) Scenario 3

5. Conclusion

In traditional cloud scheduling algorithms, the schedulable task is assigned either to the resource which finishes it in the minimum completion time or to the available resource as soon as it arrives without taking in to consideration the cost of the tasks. Some other algorithms do take into account the priority or the cost of the task assigned by the user but ignored the completion time. In this paper, we presented the (PCA) algorithm to consider both the completion time and the cost-priority in order to optimize the resource utilization and to minimize the makespan in order to provide cost economical services with high performance for the cloud users. Many issues remain open, like temperature resources and energy consumption etc., and they are under consideration as a part of a further work.

References

1.Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Special Publication 800-145.National Institute of Standards and Technology. U.S. Department of Commerce.(2011). [Online]. Available:http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

2.Sarada, N. S.: Enhanced Ant Colony System Based On RASA Algorithm In Grid Scheduling. International Journal of Computer Science and Information Technologies, Vol. 2.No.4, 1659-1674, 2011.

3.Chen, H., Wang F., Helian, N., Akanmu, G.: User-Priority Guided Min-Min Scheduling Algorithm for Load Balancing in Cloud Computing. In Proceedings of the National Conference on Parallel Computing Technologies (PARCOMPTECH), IEEE.Computer Society, Bangalore, India. (2013).

4.Beloglazov, A., Abawajy, J., Buyya, R.: Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing, The International Journal of Grid Computing and eScience, Future Generation Computer Systems (FGCS), Volume 28. No. 5, 755-768. Elsevier Science, The Netherlands. (2012).

5. Kaur, A., Kinger, S.: Temperature Aware Resource Scheduling in Green Clouds. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI).IEEE Computer Society, Mysore, India, 1919-1923. (2013).

6.Cao, Q., Wei. Z., Gong, W. M.: An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing. In Proceedings of the 3rd International Conference Bioinformatics and Biomedical Engineering (ICBBE). IEEE Computer Society, Beijing, China, 1-3. (2009).

7. Etminani, K., Naghibzadeh, M.: A Weighted Mean Time Min-Min Max-Min Selective Scheduling Strategy for Independent Tasks on Grid. In Proceedings of the 2nd International Advance Computing Conference (IACC), IEEE Computer Society, Patiala, India, 4-9. (2010)

8.Casavant, T., Kuhl, J. G.: A Taxonomy of Scheduling in General-purpose Distributed Computing Systems. IEEE Transactions on Software Engineering, Vol.14, No. 2, 141-154. (1988).

9. Bhoi, U., Ramanuj, P.N.: Enhanced Max-Min Task Scheduling Algorithm in Cloud Computing. International Journal of Application or Innovation in Engineering & Management (IJAIEM), Vol. 2, No. 4, 259-264. (2013).

10. Liu, G., Li, J., Xu, J.: An Improved Min-Min Algorithm in Cloud Computing. In Proceedings of the International Conference of Modern Computer Science and Applications. Springer, Wuhan, China, 47-52. (2012).

11. Selvarani, S., Sadhasivam, G. S.: Improved Cost-based Algorithm for Task Scheduling in Cloud Computing. In Proceedings of International Conference Computational Intelligence and Computing Research (ICCIC), IEEE Computer Society, Coimbatore, India,1-5.(2010)..

12. Ghanbari, S., Othman, M.: A Priority-based Job Scheduling Algorithm in Cloud Computing. In Proceedings of the International Conference on Advances Science and Contemporary Engineering (ICASCE). Jakarta, Indonesia, 778-785. (2012).

13. Wu, X., Deng, M., Zhang, R, Zeng, B., Zhou, S.: A Task Scheduling Algorithm Based on QoS-Driven in Cloud Computing. In Proceedings of the International Conference on Information Technology and Quantitative Management. Elsevier Procedia, Suzhou, China, 1162-1169. (2013).

14. Lee1, Z., Wang, Y., Zhou, W.: A Dynamic Priority Scheduling Algorithm on Service Request Scheduling in Cloud Computing. In Proceedings of the International Conference on Electronic & Mechanical Engineering and Information Technology (EMEIT).IEEE Computer Society, Harbin, China, 4665-4669. (2011).

15. Sun Microsystems: Take Your Business To A Higher Level - Sun Cloud Computing Technology Scales Your Infrastructure to Take Advantage of New Business Opportunities. Technical Report. (2009).