Clustering based Max-Min Scheduling in Cloud Environment

Zonayed Ahmed
Department of CSE
Stamford University Bangladesh
Dhaka, Bangladesh

Adnan Ferdous Ashrafi
Department of CSE
Stamford University Bangladesh
Dhaka, Bangladesh

Maliha Mahbub
Department of CSE
Stamford University Bangladesh
Dhaka, Bangladesh

***Abstract*—Cloud Computing ensures Service Level Agreement (SLA) by provisioning of resources to cloudlets. This provisioning can be achieved through scheduling algorithms that properly maps given tasks considering different heuristics such as execution time and completion time. This paper is built on the concept of max-min algorithm with and unique proposed modification. A novel idea of clustering based max-min scheduling algorithm is introduced to decrease overall make-span and better VM utilization for variable length of the tasks. Experimental analysis shows that due to clustering, it provides better result than the different variations of max-min as well as other heuristics algorithm in terms of effective utilization of faster VMs and proper scheduling of tasks considering all possible scheduling scenarios and picking up the best solution.**

Keywords*—Cloud computation; cluster; heuristics; batch-mode heuristics; cluster based max-min scheduling

I. Introduction

Task scheduling is a mapping mechanism from user's tasks to the appropriate selection of resources and its execution. Compared with grid computing, cloud computing has many unique features including virtualization and flexibility. By using the technology of virtualization, all physical resources are virtualized and transparent for users. All users have their own virtual device, these devices do not interact with each other and they are created based on users' requirements. In addition, one or more virtual machines can run on a single host computer so that the utilization rate of resources has been effectively improved. The independence of users' application ensures the system's security of information and enhances the availability of service [1]. Supplying resources under the cloud computing environment is flexible, we increase or reduce the supplying of resources depends on users' demand. Because of these new features, grid computing, the original task scheduling mechanism, can't work effectively in cloud computing environments [2].

The task scheduling goals of Cloud computing is providing optimal tasks scheduling for users, and provide the entire cloud system throughput and QoS at the same time. Specific goals are load balance, quality of service (QoS), economic principle, optimal operation time and system throughput [3], [4].

Task scheduling algorithm is responsible for mapping jobs submitted to cloud environment onto available resources in such a way that the total response time, the make-span, is minimized [5]. Many task scheduling algorithms are applied by resources manager in distributed computing to optimally allocate resources to tasks [6]. While some of these algorithms try to minimize the total completion time. Where the minimization is not necessarily related to the execution time of each single task, but the aim is to minimize overall the completion time of all tasks [7].

Now, for flexible resource allocation, there must be a provisioning that all resources are made available to the tasks and this is done according to SLA (Service Level Agreement) with help of parallel processing. Due to different combinations of theses SLA objectives, optimal mapping of workload to resources is found to be NP-hard [8].

The paper focuses on provisioning of a full batch of cloudlets. While other researches focus on only achieving minimal make-span, this novel idea also introduces better VM utilization through clustering the cloudlets before allocating. The novel idea of dividing and existing batch of tasks into smaller clusters is introduced in this paper. This idea along with more effective scheduling algorithm provisioned for each of the clusters helps enormously in proper scheduling of tasks to VMs which are proved spontaneously in Section 3 and Section 4 titled Proposed Methodology and Experimental Result section of this paper. The effectiveness of the newly proposed algorithm is established in the Section 5 of result comparison with the existing algorithms as described in Section 2 titled Related Works.

II. Related Works

Many heuristics have been proposed to obtain semi-optimal match. Existing scheduling heuristics can be divided into two categories: **on-line mode** and **batch-mode**.

A. On-line mode heuristics

A task is mapped to a machine as soon as it arrives at the scheduler. Some heuristic instances of this category follow:

1) Minimum Execution Time

Each task is assigned to the resource that performs it in the least amount of execution time, no matter whether this resource is available or not at that time [9].

2) Opportunistic Load Balancing

Each task is assigned to the resource that becomes ready after the current task being executed, without any consideration of the execution time of the task on the particular resource. If

more than one resource becomes ready at the particular time, one resource is chosen randomly [7].

B. Batch-mode heuristics

The tasks are collected into a set called *meta-task* (MT). These sets are mapped at prescheduled times called mapping events. Some instances of this category are as follows:

1) Suffrage

Suffrage [7] is based on the idea that a task should be assigned to a certain resource and if it does not go to that resource, the most it will suffer.

2) Max-Min

Max-Min assigns task with maximum expected completion time to the corresponding resource [9].

The Max-Min algorithm is given below.

Algorithm 1: Max-Min Algorithm

Step 1: For all submitted tasks in meta-task Ti
Step 2: For all resource Rj
Step 3: Compute $C_{ij} = E_{ij} + r_j$
Step 4: While meta-task is not empty
Step 5: Find the task T_m consumes maximum completion time.
Step 6: Assign task T_m to the resource R_j with minimum execution time.
Step 7: Remove the task T_m from meta-tasks set
Step 8: Update r_j for selected R_j
Step 9: Update C_{ij} for all T_i

The algorithm takes m Resources R_j (R_1, R_2, ..., R_m) and maps n tasks T_i (T_1, T_2, ..., T_n) on these resources. Expected execution time E_{ij} of task T_i on resource R_j is defined as required time of resource R_j to finish task T_i provided that R_j has no load when assignment occurs.

On the other side, expected completion time C_{ij} of task T_i on resource R_j is defined as the overall time consumption till finishing any assigned task previously assigned. Assume r_j denote the beginning of execution task T_i. From previous mentions, it can be concluded that $C_{ij} = E_{ij} + r_j$.

The make-span of complete schedule is defined as Max (C_i) where C_i is the completion time for a task T_i [5].

Here task T_m has maximum expected completion time and it is chosen to be assigned for corresponding resource R_j that provides minimum execution time.

Make-span is defined as a measure of the throughput of the heterogeneous computing system; like the Cloud Computing environment [9], [10].

3) Min-Min

Min-Min assigns task with minimum expected completion time to the corresponding resource [9].

4) QoS Guided Min-Min

QoS Guided Min-Min [11] adds a QoS constraint (QoS for a network by its bandwidth) to basic Min-Min heuristic. The basic idea of this procedure is that some tasks may require high network bandwidth but others can be satisfied with low network bandwidth. Thus, it assigns tasks with high QoS request first according to Min-Min heuristic.

5) QoS priority grouping scheduling

QoS priority grouping scheduling is similar to QoS guided Min-min. It is proposed by F. Dong et al. [12]. The algorithm considers two major factors: a) deadline and acceptation rate of the tasks; and b) makespan of the whole system for task scheduling. Compared to Min-min and QoS guided Min-min, it achieves better acceptance rate and completion time.

6) Segmented Min-Min

In Segmented Min-Min heuristic described in [13] tasks are first ordered by their expected completion times. Then the ordered sequence is segmented and finally it applies Min-Min to these segments. This heuristic works better than Min-Min when length of tasks are dramatically different by giving a chance to longer tasks to be executed earlier than where the original Min-Min is adopted.

7) Improved Max-Min

In Improved Max-min algorithm largest job is selected and assigned to the resource which gives minimum completion time [14].

8) Enhanced Max-Min

Here, a task just greater than average execution time is selected and assigned to the resource which gives minimum completion time [15].

9) Resource Aware Scheduling Algorithm

The algorithm presented in [16] is a combination of max-min and min-min. The algorithm covers the disadvantages of both algorithms and uses the advantages.

10)Reliable Scheduling Distributed in Cloud

RSDC [17] is another batch-mode scheduling process that uses processing time as scheduling factor. It subtracts the request and acknowledges time from the ultimate time in each processor.

The organization of this paper is as follows. In Section 3 (**Batch-mode Algorithm**), detailed explanation of any modifications of max-min will be provided. In Section 4 (**Implementation and Experiments**), we will present the implementation of our algorithm through CloudSim and analysis of our findings. Discussed in Section 4 (**Conclusion**) is a summary of our full work as well as concerns to address for the future.

III. Proposed Methodology

Reviewing max-min and other batch-mode heuristics algorithm, it can be seen, the tasks are always allocated according to their respective lengths or task sizes. Now max-min works best, but there are few long tasks and many short tasks. Because, the long task can be executed in one resource while the short tasks can concurrently run on other resources. But the max-min algorithm doesn't work well in case of variable length cloudlets. To overcome this problem, we use the idea of clustering in our proposed method. If we can create some groups of cloudlets based on their characteristics, then we can try to allocate those groups according to different

SLAs. In this paper, cloudlet length has been used to create clusters. The number of clusters can be the number of resources. Clusters can be created in different approaches such as K-means clustering algorithm [18], CURE [19], FCM [20]. Here we use standard deviation of the cloudlet lengths to create the clusters.

Next each cluster is processed separately to simulate which cluster takes the highest time of operation. This process gives a cluster enough priority to be completed first given that there are different lengths of cloudlets in the whole batch.

After simulation of each cluster the cluster consuming highest time is scheduled to the VMs using the improved max-min algorithm. Subsequently the cluster with the next highest time consuming is scheduled on the VMs. This process goes on until there are no clusters left to be scheduled.

The proposed algorithm is as follows:

Algorithm 2: Proposed Algorithm for Cluster based Max-Min Scheduling algorithm

1. Populate list of tasks T
2. Find average length of Tasks
3. Find Standard Deviation of Tasks
4. Find number of clusters in standard deviation by dividing the standard deviation in VM number of parts
5. Place each Task in the list T to specific cluster by finding minimum distance of cluster standard deviation and task length
6. Simulate each Task Cluster to find out highest make-span cluster.
7. Choose the cluster with highest make-span among the batch of the clusters
    a. For all submitted tasks in meta-task T_i
    b. For all resource R_j
    c. Compute E_{ij} based on cloudlet lengths and VMs
    d. Compute $C_{ij} = E_{ij} + r_j$
    e. While meta-task is not empty
    f. Find the task T_m consumes maximum execution time.
    g. Assign task T_m to the resource R_j with minimum completion time.
    h. Remove the task T_m from meta-tasks set
    i. Update r_j for selected R_j
    j. Update C_{ij} for all T_i
8. If there are unprocessed clusters in the batch go to step 7.
9. End Algorithm.

C. Flowchart of the Proposed Algorithm

The above flowchart in Fig. 1 shows the stepwise process of the algorithm. A simulation of the given algorithm is shown below with a given scenario.

D. Scenario for Simulation

Suppose we have 12 cloudlets to be scheduled to the VMs. The respective lengths of the cloudlets are as follows:

{1100,100,110,120,130,140,150,160,170,180,200,800}

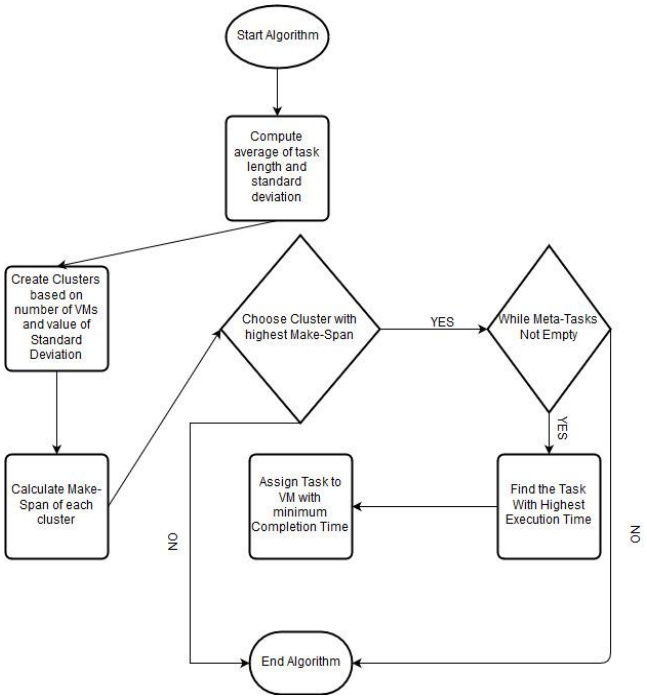

Fig. 1. Flowchart of proposed algorithm.

And the three VMs in our scenario have highest allocable MIPS as follows:

{300,100,50}

All of the VMs in the scenario have 1 core processor, 1000 Mb bandwidth, 512 Mb of RAM.

Now the total process of allocation of the tasks to the VMs is simulated in the experimental results section.

IV. Experimental Results

A. Calculation of Average and Standard Deviation of Tasks

The average length of the tasks is calculated using the simple formula:

$$ave(Length) = \frac{\sum_{i=1}^{s} Length_i}{s} \quad (1)$$

Thus the average in our scenario is: 280

Standard deviation can be calculated using the following formula:

$$S.D. = \sqrt{\frac{\sum_{i=1}^{s}(Length_i - ave(Length))^2}{s}} \quad (2)$$

Where s = number of cloudlets and $Length_i$ is the specific length of the cloudlet, i.e. the number of instructions for that specific cloudlet.

Thus the standard deviation of the given scenario would be: 307.083051.

B. Creating Clusters on the Basis of Standard Deviation

Now we need to divide our sample tasks to create clusters that would be scheduled to the VMs. According to our given scenario we are creating three clusters because we have three

VMs. If the number of VMs increases, so does our number of clusters. Thus we divide our SD in three equal parts.

1st Cluster Standard Deviation: 102.361017

2nd Cluster Standard Deviation: 204.722034

3rd Cluster Standard Deviation: 307.083051

So we determine from the task sizes which tasks have the least distance from the standard deviations. According to the given scenario, the clusters are:

Cluster 1: Task no. 2,3,4,5,6,7

Cluster 2: Task no. 8,9,10,11

Cluster 3: Task no. 1,12

We now have three clusters those have similar sized tasks within themselves. We are ready to simulate how much time the three clusters need to finish by calculating their estimated execution time, completion time and waiting times.

C. Calculation of Estimated Makespan for Each Cluster

Now we simulate each cluster to see which one gives us the maximum time make-span. We will schedule the clusters that have the highest make-span and remove all tasks of that cluster from our set of cloudlets.

For our given scenario the time make-span for each cluster along with the definitive start time, time of execution, finish time along with the VM id at which the task was executed which was determined with the help of CloudSim are followed in Table 1.

TABLE I. SIMULATION OF EACH OF THE CLUSTERS

Cluster	Cloudlet ID	VM ID	Start Time	Time	Finish Time
1	4	0	0.1	0.4	0.5
	5	0	0.5	0.43	0.93
	3	1	0.1	1.09	1.19
	6	0	0.93	0.47	1.4
	7	0	1.4	0.5	1.9
	2	2	0.1	2	2.01
2	8	0	0.1	0.53	0.63
	10	0	0.63	0.6	1.23
	9	1	0.1	1.69	1.79
	11	0	1.23	0.67	1.9
3	1	0	0.1	3.67	3.77
	12	0	3.77	2.67	6.43

Now we would choose the cluster for scheduling which has the highest make-span among all three clusters. We will go on selecting the highest cluster until all clusters are scheduled.

Thus we would process cluster 3(highest make-span 6.43 seconds) first, cluster 1(highest make-span 2.01 seconds) second and lastly cluster 2(highest make-span 1.9 seconds).

D. Scheduling of tasks of a cluster

Algorithm 3: Cluster Based Max-Min Scheduling Algorithm for each cluster

Step 1: For all submitted tasks in meta-task T_i
Step 2: For all resource R_j
Step 3: Compute E_{ij} based on cloudlet lengths and VMs
Step 4: Compute $C_{ij} = E_{ij} + r_j$
Step 5: While meta-task is not empty
Step 6: Find the task T_m consumes maximum execution time.
Step 7: Assign task T_m to the resource R_j with minimum completion time.
Step 8: Remove the task T_m from meta-tasks set
Step 9: Update r_j for selected R_j
Step 10: Update C_{ij} for all T_i

Next the tasks within a cluster are scheduled according to the Algorithm 3.

This algorithm ensures that a task T_i will be assigned to a new VM such that the overall make-span of all of the VMs remains to a minimum. That means the new task will be assigned to a new VM only if the make-span of the newly assigned task to the new VM is lesser than the make-span if the task was assigned rather to the previous VM.

As per the given scenario we see that cluster 3 having the highest make-span should be executed first to ensure that the fastest VM gets free faster than the other VMs. The specific reason behind this operation is because while processing each cluster the task that has the highest execution time is set to be completed as fast as it could be. Thus we are utilizing the fastest resources on the highest length cloudlets which will help immensely on properly executing larger tasks at hand rather than clogging the fastest resource with faster smaller tasks.

According to our given scenario the start time, finish time and total operation time are followed in Table 2.

TABLE II. OPERATION TIME OF NEW PROPOSED ALGORITHM

Cluster	Cloudlet ID	VM ID	Start Time	Time	Finish Time
3	1	0	0.1	3.67	3.77
	12	0	3.77	2.67	6.43
1	2	1	0.1	1	1.1
	3	2	0.1	2.2	2.3
	5	1	1.1	1.31	2.41
	6	1	2.41	1.4	3.81
	4	2	2.3	2.39	4.69
	7	1	3.81	1.5	5.31
2	9	1	5.31	1.7	7.01
	10	0	6.43	0.69	7.12
	11	0	7.12	0.67	7.79
	8	2	4.69	3.2	7.89

As we see above the cluster 3 is executed first which ends in VM 0(fastest resource) with the finish time of 6.43 seconds. This means the next task scheduled on VM 0 can start on 6.43 seconds. The other two VMs can now easily compute all of cluster 2 tasks within 5.31 seconds. As seen from the results we see that we have used the comparatively slower resources to execute faster smaller tasks which result in proper utilization of the VMs. Finally, the task scheduling ends with VM 0 having finish time 7.79 seconds, VM 1 with 7.01 seconds and VM 2 with 7.89 seconds.

V. Result Comparison

In our evaluation of the result with existing systems we would compare our results with several algorithms like Max-Min, Min-Min, Improved Max-Min and Enhanced Max-Min.

A. Result of Improved Max-Min on Given Scenario

We applied the improved max-min algorithm on the given scenario. The results from the simulation are followed in Table 3.

TABLE III. Operation Time of Improved Max-Min Algorithm

Cloudlet ID	VM ID	Start Time	Time	Finish Time
3	1	0.1	1.2	1.3
1	2	0.1	2	2.1
6	1	1.3	1.5	2.8
0	0	0.1	3.67	3.77
5	0	3.77	0.47	4.23
2	2	2.1	2.24	4.34
7	1	2.8	1.65	4.45
9	0	4.23	0.6	4.83
10	0	4.83	0.67	5.5
8	1	4.45	1.7	6.15
4	2	4.34	2.59	6.93
11	0	5.5	2.67	8.17

Comparing with this algorithm alone shows that the make-span of the new algorithm is better than the improved max-min algorithm.

A mere (8.17-7.79) = 0.38 seconds at VM 0 might not seem that good a result. But given the fact that this VM is the fastest VM in the given scenario proves that a fraction of a seconds in the most powerful VM can outperform several slower VMs in the scenario. Thus getting the most powerful VM free faster means the next batch of tasks can be scheduled to the VMs faster than any other traditional algorithms.

B. Comparison with Improved and Enhanced Max-Min

Given the same scenario the make-span for each of the algorithms are followed in Table 4.

TABLE IV. Comparison Chart of Improved, Enhanced and Proposed Algorithm

Algorithm	No. of Tasks	No. of VMs	Highest Make-Span
Enhanced Max-Min	12	3	10.63
Improved Max-Min	12	3	8.17
Cluster Based	12	3	7.89

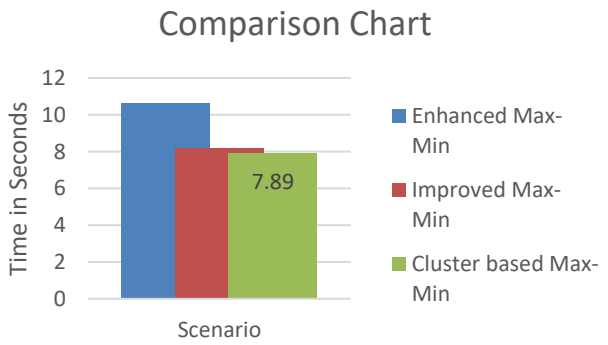

Fig. 2. Comparison Chart between traditional algorithms and proposed Cluster based max-min scheduling algorithm.

The comparison chart between the traditional algorithms (Enhanced Max-Min, Improved Max-Min) and proposed Cluster based Max-Min scheduling in shown in Fig. 2.

VI. Conclusion and Future Work

This paper concentrates on the problem of effectively scheduling tasks to VMs on a dynamic manner. The main problem of scheduling tasks in a VM is the diversity of the size of tasks that arrive for scheduling. The proposed algorithm proves to be effectively clustering the same sized cloudlets together and eventually scheduling them together. As a result, the tasks that will have the highest make-span is gotten rid of as quickly as possible ensuring that the highest VMs are freed up as soon as possible. This action results in execution of higher number of tasks in rather shorter span of time. Even if the tasks are way too much in diversity, even then this algorithm will never perform lesser than improved max-min algorithm in any situation.

On comparative analysis this algorithm can outperform any traditional algorithm on average case scenarios and no algorithm can perform better than this proposed algorithm in any worst case scenarios.

In the future other techniques (K-means clustering, Fuzzy C-means clustering) will be used for clustering and the proposed algorithm will be compared against Metaheuristic and Evolutionary algorithms to show its effectiveness. Larger dataset of cloudlets and VMs will also be used to elaborate the findings of the ongoing research.

REFERENCES

[1] Zhexi, Y.A.N.G. and Huacheng, X.U.E. 2012. Informatization Expectation with Cloud Computing in China. Indonesian Journal of Electrical Engineering and Computer Science, 10(4), pp.876-882.

[2] Liu, J., Luo, X.G., Li, B.N., Zhang, X.M. and Zhang, F., 2013. An intelligent job scheduling system for web service in cloud computing. Indonesian Journal of Electrical Engineering and Computer Science, 11(6), pp.2956-2961.

[3] You, X., Chang, G. and Deng, X., 2006. et. Grid Task Scheduling Algorithm Based on Merit Function. Computer Science, 33(6).

[4] Yao, W., Li, B. and You, J., 2002. Genetic scheduling on minimal processing elements in the grid. AI 2002: Advances in Artificial Intelligence, pp.465-476.

[5] Parsa, S. and Entezari-Maleki, R., 2009. RASA: A new task scheduling algorithm in grid environment. World Applied sciences journal, 7(Special issue of Computer & IT), pp.152-160.

[6] Chunlin, L. and Layuan, L., 2006. QoS based resource scheduling by computational economy in computational grid. Information Processing Letters, 98(3), pp.119-126.

[7] Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D. and Freund, R.F., 1999. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of parallel and distributed computing, 59(2), pp.107-131.

[8] J. M. Wilson, "An algorithm for the generalized assignment problem with special ordered sets," Journal of Heuristics, 11(4):337–350, 2005.

[9] Freund, R.F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J.D. and Mirabile, F., 1998, March. Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet. In Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh (pp. 184-199). IEEE.

[10] Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D. and Freund, R.F., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed computing, 61(6), pp.810-837.

[11] He, X., Sun, X. and Von Laszewski, G., 2003. QoS guided min-min heuristic for grid task scheduling. Journal of Computer Science and Technology, 18(4), pp.442-451.

[12] Dong, F., Luo, J., Gao, L. and Ge, L., 2006, October. A grid task scheduling algorithm based on QoS priority grouping. In Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference (pp. 58-61). IEEE.

[13] Wu, M.Y., Shu, W. and Zhang, H., 2000. Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems. In Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th (pp. 375-385). IEEE.

[14] Elzeki, O.M., Reshad, M.Z. and Elsoud, M.A., 2012. Improved max-min algorithm in cloud computing. International Journal of Computer Applications, 50(12).

[15] Bhoi, U. and Ramanuj, P.N., 2013. Enhanced max-min task scheduling algorithm in cloud computing. International Journal of Application or Innovation in Engineering and Management (IJAIEM), 2(4), pp.259-264.

[16] Parsa, Saeed, and Reza Entezari-Maleki. "RASA: A new task scheduling algorithm in grid environment." World Applied sciences journal 7.Special issue of Computer & IT (2009): 152-160.

[17] Delavar, Arash Ghorbannia, et al. "RSDC (reliable scheduling distributed in cloud computing)." International Journal of Computer Science, Engineering and Applications 2.3 (2012): 1.

[18] Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics) 28.1 (1979): 100-108.

[19] Guha, Sudipto, Rajeev Rastogi, and Kyuseok Shim. "CURE: an efficient clustering algorithm for large databases." ACM Sigmod Record. Vol. 27. No. 2. ACM, 1998.

[20] Bezdek, James C., Robert Ehrlich, and William Full. "FCM: The fuzzy c-means clustering algorithm." Computers & Geosciences 10.2-3 (1984): 191-203.