ISSN:0975-9646

An Improved Heft Algorithm Using Multi-Criterian Resource Factors

Renu Bala
M Tech Scholar, Dept. Of CSE ,
Chandigarh Engineering College,
Landran , Mohali , Punajb

Gagandeep Singh
Assistant Professor, Dept. Of CSE ,
Chandigarh Engineering College,
Landran , Mohali ,Punjab

***Abstract*- Since the HEFT algorithm primarily work on basics of the calculated earliest time in execution from the available resources, it ignores the factors that affect the time in execution of tasks. The proposed algorithm includes multiple new factors for matching right resources for particular task to be scheduled. These factors include inter-node bandwidth between VM nodes, storage, and RAM requirement for execution of task. The proposed algorithm shows that the algorithm is able to reduce the turnaround time, as well as the waiting and execution time due to correct selection of resources dependent on the RAM, bandwidth and storage factors.**

***Index Terms*- Scheduling, Cloud Computing, HEFT Algorithm.**

I. INTRODUCTION

Scheduling refers to the set of policies to control the order of work to be performed by a cloud. Scheduling policies in a cloud environment vary depending on the deployment model of the cloud. Scheduling manages availability of cloud/resource grid and good scheduling policy gives maximum utilization of resource [13]. Job scheduling is one of the major activities performed in all the computing environments. Cloud computing has tremendous capabilities and to make effective use of limitless capabilities efficient scheduling algorithms are required as cloud computing is one of the upcoming latest technology which is developing drastically. Scheduling strategy is the key technology in cloud computing [7]. The scheduling process in cloud can be summarized into three stages:

1) Resource discovery and filtering: The resources present in the network system are discovered and information related to them is collected.
2) Resource selection: The target resource is selected based on certain parameters of task and resource.
3) Task submission: The task to be executed is submitted to resource selected.

The goal of scheduling algorithms in distributed systems is spreading the load on processors and maximizing their utilization while minimizing the total execution time [6]. There are two main categories of scheduling algorithm:
1. Static scheduling algorithm
2. Dynamic scheduling algorithm

Both have their own advantages and limitations. However, Dynamic scheduling algorithm but has a lot of overhead compare to it. In static scheduling, optimal resource allocation of activities over time is done and all resources and all activities are given. There is no uncertainty in the behavior of resources and activities [11].
In a cloud, there are four main entities and that are cloud user, Broker, Virtual Machines and Physical Machines. The cloud users are the actual consumers of services and can submit their service requests from anywhere in the world. A cloud datacenter comprises of physical machines. By utilizing virtualization technology, virtual machines are created on the top of physical machines. Broker acts as an intermediator between cloud users and cloud datacenters. It is responsible for allocating cloud resources to client's work flow applications [14].
There have been various types of scheduling algorithm existing in distributed computing system. The scheduling algorithms provide benefit to both, the cloud user as well as the service provider [8]. Scheduling algorithms can be designed in the following ways:-

1. It can be designed in such a way that they satisfy the Quality of Service (QoS) constraints imposed by cloud users.
2. It can be designed to perform load balancing among virtual machines which results into improvement of resource utilization at service provider's end.

II. RELATED WORK

There have been various types of scheduling algorithm existing in distributed computing system. Traditional job scheduling algorithms are not able to provide scheduling in the cloud computing tended to use the direct tasks of users as the overhead application base. There is an urgent need to develop new scheduling strategies that may use some of the conventional scheduling concepts to merge them together with some network aware strategies to provide solutions for better and more efficient job scheduling for next generation of cloud.
Bittencount Luiz F. et. al. [1] provides an improvement of Heterogeneous Earliest Finish Time (HEFT) which does not consider the estimates o a single task for the locally optimal decisions but also look ahead in the schedule and consider the information that effect the children of the task by the decisions made. The key idea of this paper is to improve the process of scheduling tasks in HEFT, by looking ahead and considering information about the descendants of a task.

Zhao H. et. al. [11] considered Heterogeneous Earliest Finish Time (HEFT) algorithm for scheduling the tasks of an application, represented by a directed acyclic graph, onto a bounded number of heterogeneous machines. In this paper, a number of different options for computing the weights in HEFT are considered. In HEFT, a weight is allocated to each node and edge of the graph, based on the average computation and communication, respectively. At that point, the graph is traversed upwards and a rank value is assigned to each node. Tasks are then scheduled, in order of their rank value, on the machine which gives the earliest finish time.

Topcuoglu H. et. al. [8] presents low-complexity efficient heuristics known as Heterogeneous Earliest-Finish-Time (HEFT) algorithm and the Critical –Path-on-a-Processor (CPOP). These scheduling algorithms are used for a bounded number of heterogeneous processors with an objective to simultaneously meet high performance and fast scheduling time. The HEFT algorithm selects the task with the highest upward rank value at each step and assigns the selected task to the processor, which minimizes the earliest time with an insertion-based approach. But the CPOP algorithm uses the summation of upward and downward rank values for prioritizing tasks.

Cui Lin et. al. [3] proposed a new workflow scheduling algorithm to schedule a workflow elastically on a cloud computing environment which is known as SHEFT (scalable Heterogeneous Earliest-Finish-Time) algorithm. As per this paper, In a cloud computing environment, even though the number of assigned resources to a workflow can be elastically scaled, there exists some scheduling problems such as the number of resources cannot be automatically determined on demand of the size of a workflow and the resources assigned to a workflow are not released until the workflow completes an execution. To solve these scheduling problems, Scalable heterogeneous Earliest-Finish-Time (SHEFT) is proposed.

Selvarani S. et. al. [8] proposed an Improved Cost-Based scheduling Algorithm in this paper. The improved Cost-Based Scheduling Algorithm in cloud computing is employed for making efficient mapping of tasks to available resources in the cloud. This algorithm selects a set of resources to be used for computing. The results show that the processing cost spent to complete tasks after grouping the tasks is very less when compared with the processing cost spent to complete the tasks without grouping the tasks.

Chen Weiwei et. al. [2] introduces Workflowsim simulator, which extends the existing CloudSim simulator by providing a higher layer of workflow administration. In this paper, new workflow simulator is introduced which takes into consideration heterogeneous system overheads and failures that existing workflow simulators fail to provide.

Long W. et. al. [6] introduces a simulation framework called CloudSim which provides simulation, power to manage services and modeling of cloud infrastructure. Cloud computing will be a major technology in the development of the future Internet of Services. Service providers want to remove the bottle neck of the cloud computing system in order to satisfy user requirement.

III. EXISTING ALGORITHM

The HEFT Algorithm is an application scheduling algorithm for a bounded number of heterogeneous processors. The algorithm first constructs a priority list of tasks and then locally optimal allocation decisions for each task are made on the basis of the task's estimated finish time. The objective of efficient scheduling is to map the tasks onto the core processors and execution order is set so that task precedence requirements are satisfied and minimum schedule length is given. The HEFT algorithm is an effective solution for the DAG scheduling problem on heterogeneous systems because of its robust performance, low running time, and the ability to give stable performance over a wide range of graph structures. The limitation of HEFT algorithm is that it uses techniques that are all static approaches of the mapping problem that assume static conditions for a given period of time and also in complex situations it can easily fail to find the optimal scheduling. [2][17].

1) The HEFT algorithm first calculates average execution time for each task and average communication time between resources of two successive tasks. Let time (T_i, r) be the execution time of task T_i on resource r and let R_i be the set of all available resources for processing T_i. The average execution time of a task T_i is defined as

$$\overline{\omega} = \frac{\sum_{r \in R_i} \mathrm{time}(T_i, r)}{|R_i|} \qquad (3.1)$$

2) Let time (e_{ij} , r_i, r_j)be the data transfer time between resources r_i and r_j which process the tasks T_i and T_j respectively. Let R_i and R_j be the set of all available resources for processing T_i and T_j respectively. The average transmission time T_i to T_j is defined by:

$$\overline{c_{ij}} = \frac{\sum_{r_i \in R_i, r_j \in R_j} \mathrm{time}(e_{ij}, r_i, r_j)}{|R_i|\,|R_j|} \qquad (3.2)$$

3) Then tasks in the workflow are ordered in HEFT based on a rank function. For a exit task T_i, the rank value is:

$$\mathrm{rank}\,(T_i) = \overline{\omega}_i \qquad (3.3)$$

4) The rank values of other tasks are computed as:

$$\mathrm{rank}\,(T_i) = \overline{\omega_i} + \max_{T_j \in succ(T_i)}(\overline{c_{ij}} + \mathrm{rank}\,(T_i)) \qquad (3.4)$$

where, *succ(T_i)* is the set of immediate successors of task T_i. The algorithm then sorts the task by decreasing order of their rank values. The task with higher rank value is given higher priority. In the resource selection phase, tasks are scheduled according to their priorities and each task is assigned to the resource that can finish the task at the earliest time [19].

Algorithm 1. Heterogeneous –Earliest Finish-Time (HEFT) algorithm

```
1: compute the average execution time for each task $t \in \Gamma$ according to equation 3.1
2: compute the average data transfer time between tasks and their successors according to equation 3.2
3: compute rank value for each task according to equation 3.3 and 3.4
4: sort the tasks in a scheduling list Q by decreasing order of task rank value
5: while Q is not empty do
```

```
6: t ← remove the first task from Q
7: r ← find a resource which can complete t as earliest time
8: schedule t to r.
9: end while
```

IV. PROPOSED ALGORITHM

The proposed algorithm is an improvement of the HEFT algorithm. The HEFT algorithm is used to schedule many kinds of tasks which have been put in a workflow, and these tasks have different requirements in terms of resources for successful execution. The goal of HEFT is to minimize the workflow makespan but it does not consider factors such as inter-node bandwidth, RAM and storage memory. These factors are included in the proposed algorithm for efficient working of the scheduling algorithm and for better results.

1) Score of Request, S_R is calculated which is

$$S_R = \frac{a}{b} * \frac{c}{d}$$

where, *a* is get_BW_Reuested

b is get_BW_internode_congestion_list

c is get_RAM_Request

d is get_Storage_Request

2) Score of Available, S_A is calculated which is

$$S_A = \frac{x}{y} * \frac{z}{u}$$

where, x is get_BW_Available

y is get some value based on TTL value (ping test)

z is get_RAM_Available

u is get_Storage_Available

Algorithm 2. Proposed Algorithm

```
1: compute the average execution time for each task $t \in \Gamma$ according to equation 3.1
2: compute the average data transfer time between tasks and their successors according to equation 3.2
3: for each task in workflow, if $S_A > S_R$, then continue:
4: compute compound rank value for each task according to equation 3.3 and 3.4
5: sort the tasks in a scheduling list Q by decreasing order of task rank value
6: while Q is not empty do
7: t ← remove the first task from Q
8: r ← find a resource which can complete t as earliest time
9: schedule t to r.
10: end while
```

A. PROPOSED APPROACH

We are using application specific workloads for giving work to CloudSim simulator. This may require special treatments to tasks and its dependency in terms of its mapping the work to follow a specific flow.

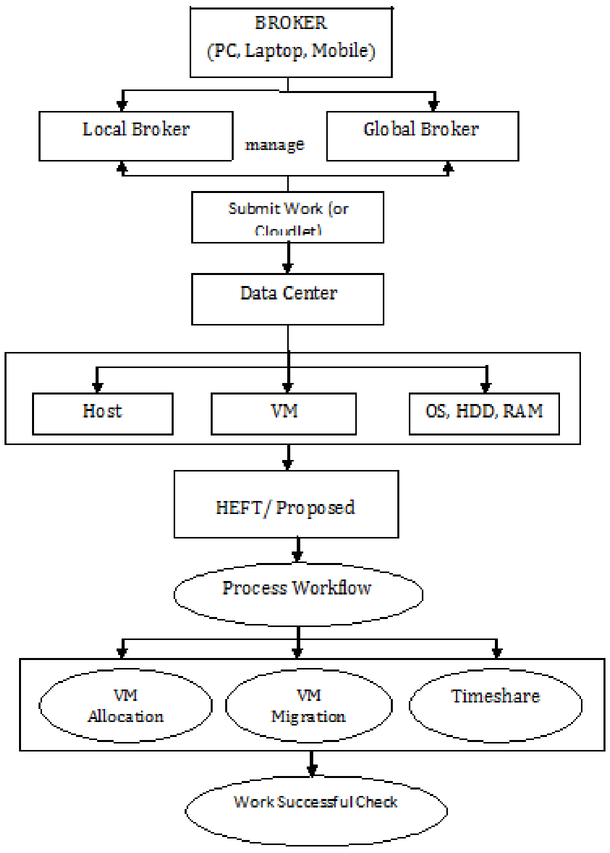

Figure 4.1 Block Diagram showing the basic working of the HEFT Algorithm and Proposed Algorithm

The development of system framework includes the following steps that are explained in detail as follows:

A. Create Local and Global Broker

A cloud broker may be a third-party individual that acts as an intermediately between the purchaser of a cloud computing service and the seller of that service.

Local Broker: A local Broker is an entity that manages the use, performance and delivery of cloud services and establishes relationship between cloud service providers and cloud service consumers.

Global Broker: A Global Broker system supports fast provisioning of resource infrastructure needed in service evaluation, system and computational resources, over the multiple clouds.

B. Create Data Center

Datacenters are the resource providers in CloudSim 3.0. Datacenter is the heart of the network cloud. Data center process all work, which is submitted by various brokers. Data center characteristics object that stores the properties of a data center: architecture, OS, list of Machines, allocation policy: time or space-shared, time zone and its price. It encapsulates a set of compute hosts that can either be homogeneous or heterogeneous with respect to their hardware configurations (memory, cores, capacity, and storage).Additionally, every Datacenter component instantiates a generalized application provisioning component that implements a set of policies for allocating bandwidth, memory and storage devices to hosts and VMs.

In order to create a Power Datacenter, firstly we need to create a list of store one or more machines. A machine contains one or more PEs or CPUs/Cores. Therefore, users should create a list to store this PEs (Processing Elements) before creating a machine. Datacenter has its own policy of how it will process the submitted work (or cloudlets).

C. Coding of HEFT Algorithm/Proposed Algorithm

The conversion of HEFT Algorithm and Proposed Algorithm from pseudo code to coding in java using NetBeans 7.0 is done for HEFT Algorithm, the tasks are created and rank of each task is defined. The task with the highest priority which is defined on the basis of rank of each task is allocated resources and the task is executed. The remaining tasks are executed in the same way.

For proposed algorithm, the bandwidth between each pair of virtual machines in the bandwidths of Parameters is specified and the aim is to optimize the communication cost instead of using the average communication cost in HEFT. In this, a scoring system based on RAM, Bandwidths and storage is build.

D. Process Workflow

The workflow is processed that is the results of the HEFT Algorithm and Proposed Algorithm is processed. The output of both the algorithms are generated and then recorded to create graphs. The graphs are created to compare these two algorithms and to show that the proposed algorithm is better which includes new parameters. At the end, by comparing graphs it is checked whether the work is successful or not.

V. RESULTS

The proposed algorithm is implemented on an Intel Core 2 Duo machine with 230 GB HDD and 3 GB RAM on 64-bit OS. The experiments are conducted on a simulated Cloud environment provided by CloudSim 3.0. The speed of each processing element is expressed in MIPS (Million Instructions Per Second) and the length of each cloudlet is expressed as the number of instructions to be executed. The algorithms are tested by varying the number of cloudlets and also randomly varying the length of cloudlets. Also, the number of VMs used to execute the cloudlets, are varied accordingly. Comparative analysis of our proposed algorithm with existing algorithm show that the proposed algorithm have better results and is more reliable scheduling algorithm. The following two parameters are considered for the result evaluation:

A) Turnaround Time

It is the total time taken between the submission of a task for execution and the return of the complete output to the user.

Turnaround Time= Submission Time + Waiting Time + Execution Time

The turnaround time is compared for the existing algorithm (HEFT) [2][17] which is without including parameters and the proposed algorithm with parameters. The comparison is done with the number of tasks and by increasing the number of tasks gradually. For any given scheduling algorithm, it is expected that the turnaround time or the total time in execution of task, from time it was submitted till it got finally executed with success flag should be minimized. The results obtained are shown in form of graph in figures:

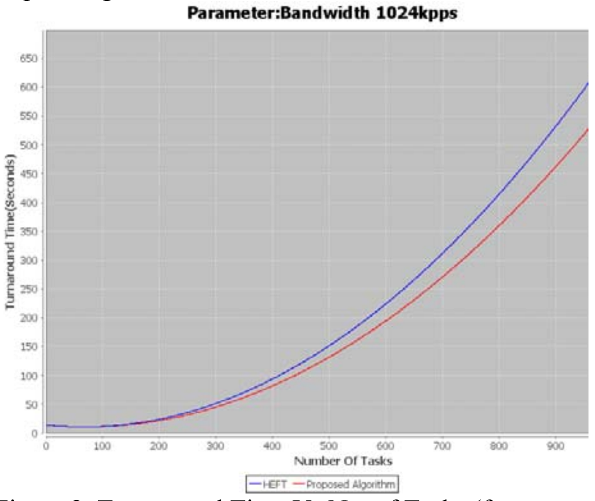

Figure 2. Turnaround Time Vs No. of Tasks (for parameter Bandwidth 1024kpps)

Analysis: The inter-node bandwidth is added as a parameter to study congestion and to implement resource dependency and task execution dependency. The results from the graph show that the turnaround time values for the proposed algorithm is less as compared to the HEFT algorithm which is required in the proposed algorithm.

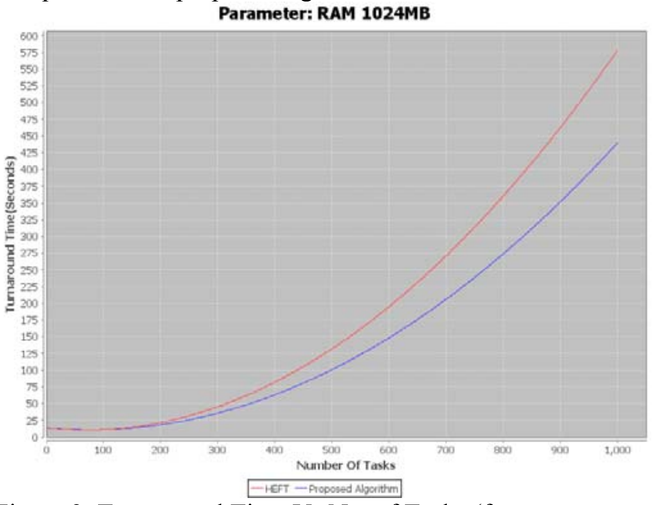

Figure 3. Turnaround Time Vs No. of Tasks (for parameter RAM 1024MB)

Analysis: If the job to be executed has data more than the size of the RAM (4 GB), then the combination of HardDisk and RAM is required and thus we have added this parameter. From the results it is shown that the values of turnaround time are less in case of proposed algorithm as compared to the HEFT algorithm. This is due to the additional RAM (i.e. 1024 MB) taken as a parameter.

B) Response Time

The response time of a task or thread is defined as the time when task is ready to execute to the time when it finishes its job.

Response Time= Arrival Time – Finish Time

The response time is compared for the existing algorithm (HEFT) [2][17] which is without including parameters and the proposed algorithm with parameters. The comparison is done with the number of tasks and by increasing the number of tasks gradually. The results obtained are shown in form of graph in figures:

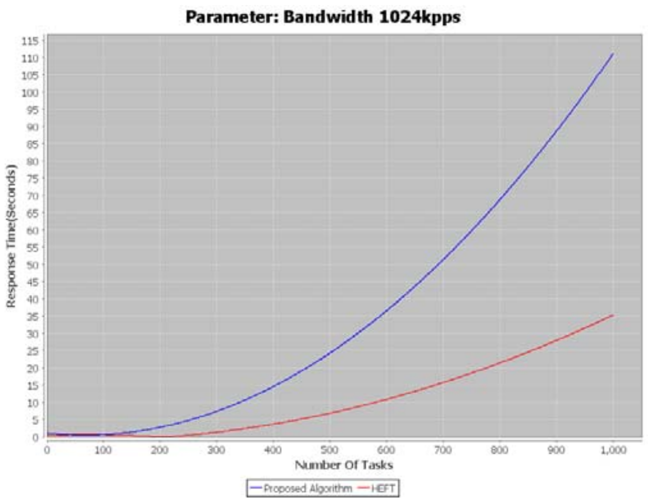

Figure 3. Response Time Vs No. of Tasks (for parameter Bandwidth 1024kpps)

Analysis: The response time depends on a variety of factors and one of them is bandwidth. The network speed can be increased by implementing inter-node bandwidth and the result from the graph concludes that the proposed algorithm has good response time.

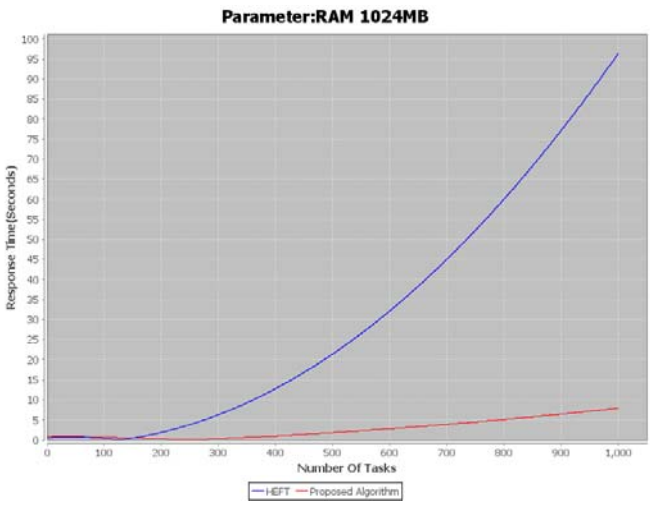

Figure 4. Response Time Vs No. of Tasks (for parameter RAM 1024MB)

Analysis: Initially, for any computational operations there is first need for RAM. This memory is first utilized; hence its availability helps the schedulers to perform better as they are able to reduce the response, execution time. In the graph, blue line represents the HEFT algorithm and red line represents the proposed algorithm and it is evident from the graph that the proposed algorithm outperforms the existing algorithm as the line is below as shown in the above graph.

VI. Conclusion And Future Work

In this research work, we studied the HEFT algorithm [2][17] and found that it has certain limitations. The research work proposed to implement their scheme. The proposed algorithm based on HEFT algorithm consider factors such as inter-node bandwidth, RAM memory, Storage, etc. Results show that proposed algorithm is reliable and by reliable we mean it checks resources and it does not mismatch resources which were not done in the HEFT algorithm. The proposed algorithm uses double ranking that is selection criteria for tasks is based on execution time and resources. Comparative analysis of our proposed algorithm with existing algorithm [2][17] show that the proposed algorithm have better results and is more reliable scheduling algorithm. The future work to be carried out under the current research work should includes workload that is partitioned or divided before scheduler takes on the workload for scheduling. There are existing partition algorithm which implement clustering to achieve this goal, however for future scope, we suggest collaborative clustering approach to do new type of task partitioning. By doing this, the underlying network of sub tasks are also considered for taking partitioning decisions rather than taking decision on the basis of Root Node characters only.

References

[1] Bittencourt, L.F., Sakellariou, R., and Madeira, E.R.M. (2010), *"DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm"*, 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Publication IEEE Conference, 17-19 Feb. 2010, Pisa, pp. 27-34.

[2] Chen, W. and Deelman, E. (2012), *"WorkflowSim: A toolkit for simulating scientific workflows in distributed environments",* 8th International Conference on E-Science, IEEE Publication, 8-12 Oct. 2012, Chicago, IL, pp. 1-8.

[3] Cui Lin and Shiyong Lu (2011), *"Scheduling Scientific Workflows Elastically for Cloud Computing",* IEEE International Conference on Cloud Computing on Cloud Computing (CLOUD), 4-9 July 2011, Washington, DC, pp. 746-747.

[4] Dubey, S., Jain, V., and Shrivastava, S. (2013), *"An Innovative Approach for Scheduling of Tasks in Cloud Environment",* Fourth International Conference on Computing, Communications and Network Technologies (ICCCNT), IEEE Publication, 4-6 July 2013, Tiruchengode, pp. 1-8.

[5] Fang, Y., Wang, F., and Ge, J. (2010), *"A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing",* Web Information Systems and Mining Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, Vol. 6318, pp. 271-277.

[6] Long, W., Yuqing, L., and Qingxin, X. (2013), *"Using CloudSim to Model and Simulate Cloud Computing Environment",* 9th International Conference on Computational Intelligence and Security (CIS), IEEE Publication, 14-15 Dec. 2013, Leshan, pp. 323-328.

[7] Meng Xu, Lizhen Cui, Wang, H., and Yanbing Bi (2009), *"A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing",* IEEE International Symposium on Parallel and Distributed Processing with Applications, 10-12 Aug. 2009, Chengdu, pp. 629-634.

[8] Selvarani, S. and Sadhasivam, G.S. (2010), *"Improved cost-based algorithm for task scheduling in cloud computing",* IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 28-29 Dec. 2010, Coimbatore, pp. 1-5.

[9] Shah, S.N.M., Bin Mahmood, A.K., Oxley, A., and Zakaira, M. N. (2012), *"QoS based performance evaluation of grid scheduling algorithms",* International Conference on Computer & Information Science (ICCIS), IEEE Publication, 12-14 June 2012, Kuala Lumpeu, pp. 700-705.

[10] Topcuoglu, H., Hariri, S. and Min-You Wu (2002), *"Performance-effective and low-complexity task scheduling for heterogeneous computing"*, IEEE transactions on Parallel and Distributed Systems, Vol. 13, No. 3, pp. 260-274.

[11] Zhao, H. and Sakellariou, R. (2008), *"An Experimental Investigation into the Rank Function of the Heterogeneous Earliest Finish Time Scheduling Algorithm"*, Euro-Par 2003 Parallel Processing , Springer Berlin Heidelberg, Vol. 2790, pp. 189-194.