International Journal of Advanced Science and Technology
Vol. 28, No. 17, (2019), pp. 178-191

Performance Evaluation of Enhanced Heterogeneous Earliest Finish Time Algorithm for DAG Task Scheduling in Cloud Computing

K. Nithyanandakumari, S.Sivakumar

Abstract

Cloud environment provides enormous measure of computing, storage and networking resources. Managing and providing process resources to extensive number of users and execution of tremendous applications is a challenge in cloud computing. Attaining proficiency and providing fairness to tasks execution, scheduling of tasks is essential. Task scheduling refers assigning resources to every task and it becomes the essential factor in increasing the performance for the dynamic allocation of resources. The services application storage, network, server and other services can be utilized efficiently and it results reduced makespan and optimum cost. Heuristic techniques are based upon extensive search. Heterogeneous Earliest Finish Time (HEFT) algorithm is a heuristic which arranges tasks by rank function. A greater priority is given to the task with a higher rank and every task is assigned to resources according to its priority. In this research work, an Enhanced HEFT (EHEFT) heuristic algorithm is proposed with a pareto optimization operator called crowding distance to designate priorities to the equal ranked tasks. The proposed method is compared with HEFT, DHEFT, MaxMin, MCT and MinMin and obtains the quantitative measurements for makespan and total cost.

Keywords : *HEFT, Cloud Computing, DAG task scheduling, heuristic algorithm*

I. INTRODUCTION

Cloud computing is a field that is perceiving a rapid advancement both in academia and industry. Cloud is a kind of parallel and dispersed framework comprising of interconnected and virtualized computers that are dynamically provisioned and represented as cooperative computing resources. Clouds can be classified into different service types, namely Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). The deployment strategies are private, public, community and hybrid [1]. Different resources and services are offered to different users by the cloud service providers. The goal of a cloud service provider is dynamically provide the widely distributed set of services thereby maximizing the resource utilization, throughput and minimize the makespan and cost. In order to achieve these objectives, one of the most important components called scheduling is adopted in the cloud environment. Scheduling is referred to as the issue of finding appropriate allotment of tasks to resources [2].

Cloud scheduling deals with large number of processes that race to obtain resources. A process can be created as a workflow by partitioning it into smaller sub methods (ie., tasks) which may additionally have exclusive sizes and require distinctive running times. A workflow is defined as a set of tasks with dependencies between them. To demonstrate empirical and data-intensive applications, workflow model is widely used. A workflow can be described as a DAG (Directed Acyclic Graph), a graph with no cycles [3].

The overall performance and effectiveness of cloud computing service relies on the scheduling of workflows to the cloud resources. As workflow or DAG scheduling depends on optimization objectives like makespan, cost, reliability, security, load balancing etc., of scientific applications, it is most vital involved in a cloud computing environment. Therefore an effective workflow scheduling algorithm is required to optimize the related parameters. Optimizing workflow scheduling problem in cloud has been demonstrated to be NP-complete, therefore the requirement for the application of heuristic methods [4]. The heuristic techniques are constantly effective, powerful and flexible method for scheduling workflows.

An important class of heuristic technique which is otherwise known as the list scheduling algorithm can be applied to solve the workflow scheduling problem. The primary idea of list scheduling consists in keeping an ordered task listing by assigning priority for every task according to some greedy heuristics. Tasks are then selected for scheduling in the order of their priorities [5]. Several list scheduling heuristics have been evolved, but developing an appropriate method for a scheduling problem with a particular nature is difficult. Some list scheduling heuristics are: Highest Level First algorithm (HLF), Longest Path Method (LPM), Dynamic Level Scheduling (DLS), Critical Path Method (CPM) and Heterogeneous Earliest Finish Time (HEFT) algorithm.

Among these algorithms, the HEFT heuristic is well-known to generate good results with a minimum makespan. HEFT executes in two phases, each task is given a rank value in the first phase while the processor assignment is done based on its rank value in the second phase [6]. This heuristic primarily works to reduce the total execution time only. When executing a workflow schedule in cloud, minimizing the makespan as well as the total execution cost is a challenging problem. In this research work, a novel optimization of this heuristic is proposed as a means to lower the makespan and total execution cost incurred by using a set of heterogeneous resources. We optimize the task ranking with the multi-objective optimization operator crowding distance. The proposed heuristic EHEFT is to be evaluated for different scientific workflow applications and comparison is done with standard algorithms.

II. RELATED WORK

Scientific workflow scheduling of an application is a combinatorial optimization problem with distinct scheduling criteria in cloud computing. The most important criteria are the makespan and total cost. Many heuristic algorithms have been suggested for workflow scheduling. HEFT is a extensively approved heuristic that schedules a workflow on to a heterogeneous multiprocessor environment. Numerous upgrades and variations of the HEFT algorithm have been put forwarded for the reason.

A hybrid approach that is less sensitize toward the ranking system is proposed [7]. This heuristic is made of three steps: ranking, grouping and scheduling. In the first step, each node is being assigned a weight. Upward ranking is computed using this weight and each is node is assigned a rank value. all independent tasks are grouped together in the grouping step. In the third step, a schedule of the DAG will be obtained by considering each group in increasing order of its number, to schedule tasks in parallel.

A novel multi-objective technique referred to as multi-objective heterogeneous earliest finish time (MOHEFT) algorithm is introduced for workflow scheduling in Amazon EC2 [8]. In each step, MOHEFT builds numerous parallel workflow schedules at halfway point rather than a single schedule. To make sure the diversity of tradeoff solutions, MOHEFT used crowding distance and dominance relationships.

An extension of the HEFT algorithm called the scalable heterogeneous earliest finish time (SHEFT) that focuses to optimize the execution time of workflows and enables the resources to scale elastically at runtime [9]. All resources are partitioned into a distinct number of clusters which have the same computing capability, same network communication, and the same inter-cluster and intra cluster data transfer rate in this scheme. SHEFT is used to map a workflow on bounded number of processors after task prioritization phase.

. An extension of HEFT, the heuristic BHEFT offers a BDC (Budget and Deadline Constrained) plan [10]. While selecting the resource, the spare budget for every workflow task is considered during the formation of a BDC plan. This heuristic algorithm is only applicable to the heterogeneous computing structures where the total number of resources is absolute.

The different alterations of HEFT shows that it can be farther optimized. This research work comes up with a heuristic, based on the list based heuristic HEFT for multi-objective scheduling of workflows in a cloud. We introduce a pareto optimization operator called crowding distance which assigns priority to the equal ranked tasks with a significant cost reduction and makespan.

III. PROBLEM DEFINITION

A. System model

Workflow applications are commonly modelled as a Directed Acyclic Graph (DAG) to describe computational tasks (nodes) and their data dependencies (edges). A workflow application can be formally represented as WA=(T,E) where *T* is the set of *n* tasks $\{ta_1, ta_2, ..., ta_n\}$ *and E* is the set of edges that represent the data dependencies among them. An edge $e_{ij} = \{(ta_i, ta_j) | ta_i, ta_j \in T\}$ depict the dependency between the tasks ta_i and ta_j.

Let M $=\{m_1, m_2, m_3, \ldots\ldots m_k\}$ be the set of available virtual machines. Therefore, the workflow scheduling problem is the mapping of workflow tasks to the virtual machines $(T \rightarrow M)$.

The data flow dependency, represented by (ta_i, ta_j), indicates that there is a precedence constraint indicating that the execution of the task ta_i should be completed before the start of ta_j.Task ta_i is an immediate predecessor of task ta_j and task ta_j is an immediate successor of task ta_i. As a task may have multiple predecessors and successors, the sets $pred(ta_i)$ and $succ(ta_i)$ are used to denote the set of immediate predecessors and successors of task ta_i. That is,

$$pred(ta_i) = \{ta_j | (ta_j, ta_i) \in E\} \quad (1)$$

$$succ(ta_i) = \{ta_j | (ta_i, ta_j) \in E\} \quad (2)$$

task ta_i can start only after all tasks of *pred*(ta_i) are completed. The set $pred(ta_i)$ is called the parent set of ta_i.. All tasks of $succ(ta_i)$ will be executed after the completion of task ta_i and $succ(ta_i)$ is called the children set of ta_i.

A task ta_{entry} with no predecessors is an entry task which satisfies

$$pred(ta_{entry}) = \emptyset \quad (3)$$

while a task ta_{exit} with no successors is an exit task which satisfies

$$succ(ta_{exit}) = \emptyset \quad (4)$$

further, the execution time (ET) from/to these tasks is zero.

Table I outlines the notations and its descriptions used throughout this paper.

Table -I. Notations and Descriptions

Notations	Descriptions
T	Set of Workflow tasks
M	Set of Virtual Machines (VM)
ta_i	ith task, i=1,2,3,….p
e_{ij}	An edge which represents $\{(ta_i, ta_j) \| ta_i, ta_j \in T\}$
m_r	rth VM, r=1,2,3,…..k
ta_{entry}	A workflow's entry task
ta_{exit}	A workflow's exit task
$pred(ta_i)$	Predecessor set of ta_i
$succ(ta_i)$	Successor set of ta_i
p	Total number of tasks

q	Total number of resources/VMs
MP,f₁	Makespan of workflow
MP_i	Makespan of task *ta_i*
$AT(m_r)$	Available time of VM ***m_r*** for the execution of task *ta_i*
CC_r	Computing capacity of VM ***m_r*** measured by MIPS
f_2	Total execution cost of workflow
ST	Start time of a task
ET,F_i	Execution time of a task
FT	Finish time of a task
ω_i	Average execution time of task ***ta_i***
n	Number of successors of ***ta_i***
CT	Communication cost
CD	Crowding Distance
PCS_i	Processing cost of i^{th} task for utilizing i^{th} VM
$PCBW_i$	Processing cost for utilizing Bandwidth if i^{th} task
FS_i	File size of i^{th} task
S	The size of data needed to be communicated from task *ta_i* and *ta_j* on VM *m_r*
BW_r	Bandwidth capacity of VM *m_r*

The workflow scheduling is treated as a multi objective problem (MOP) in this work. Multi objective optimization is an area of multiple criteria decision making involving more than one objective function that requires simultaneous optimization.

B. Heterogeneous Earliest Finish Time (HEFT) Algorithm

HEFT is a list scheduling algorithm, appropriate for scheduling dynamic tasks in heterogeneous computing systems. HEFT has two states, task prioritization and processor (VM) selection states. In task prioritization state, tasks were ranked by their computation as well as communication cost. Then, all the tasks are arranged using their ranks and a higher priority is given to the task with a greater rank. In processor selection state, a suitable machine was selected for scheduling the priority based task which can complete the schedule at the earliest finish time. This state is repeated until all tasks are scheduled on proper VMs.

The ranks of the tasks are calculated by traversing the workflow in a breadth first search (BFS) manner. This is also called as the upward ranking method. The rank of workflow tasks can be computed as

$$urank_1(ta_i)=\begin{cases}\overline{\omega_i} & if\ \ succ(ta_i)=\phi\\ \overline{\omega_i}+\max\limits_{ta_j\in succ(ta_i)}\left(\overline{CT_{ij}}+urank_1(ta_j)\right) & if\ \ succ(ta_i)\neq\phi\end{cases} \tag{5}$$

$$\varpi_i=\left(\left(\sum_{i=1}^{p}ET(ta_i)\right)\Big/p\right) \tag{6}$$

$$ET(ta_i) = \frac{FS_i}{CC_r} \quad (7)$$

$$\overline{CT_{ij}} = \frac{\sum_{m_i \in M_i, m_j \in M_j} S(e_{ij}, m_i, m_j)}{|M_i||M_j|} \quad (8)$$

If two or more tasks have equal priority in the task prioritization phase, then the tie is resolved through selecting a task randomly. Fig.1 shows the number of chances to get equal priority for n number of tasks in Montage Datasets [17].

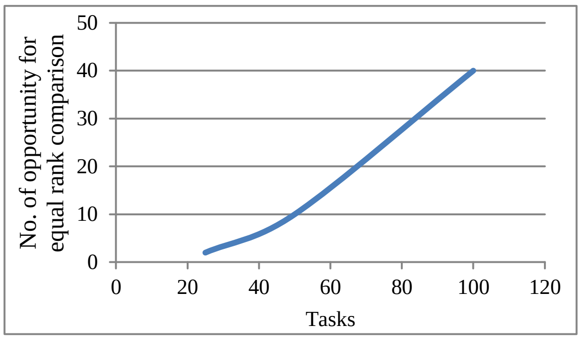


Fig. 1. Number of chances to get equal rank of tasks in Montage

C. Problem Formulation

The HEFT heuristic selects the task randomly for scheduling in the task prioritization phase when two or more tasks having the same rank. Fig.1 shows that as the number of tasks increases, the number of chances for comparing equal ranked tasks also increases. The proposed EHEFT algorithm uses a mathematical method crowding distance for assigning priority for the equal ranked tasks.

The proposed work is intended to reduce the makespan and cost. It is impossible to search out a solution that minimizes both makespan and total cost simultaneously. A solution *a* prevails a solution *b*, if the makespan and cost of *a* are smaller than those of *b*. Conversely, two solutions are said to be nondominated whenever none of them dominates the other (i.e.one is better in makespan and the other is better in cost). The set of optimal non-dominated solutions is referred to as Pareto front [12]. A solution is Pareto front if either it is at least as good as all other solutions for all the two objectives.

In the proposed workflow scheduling problem, the fitness of a solution is the tradeoff between the two objectives. The scheduling problem can be formulated as a MOP given by

$$Min\ f_w = (f_1, f_2) \quad (9)$$

where

$$f_1 = MP$$

$$f_2 = \sum_{i=1}^{p} (PCS_i * MP_i) + (PCBW_i * FS_i)$$

subject to

$$MP = FT(ta_{exit}) \quad (10)$$

$$FT(ta_i) = ST(ta_i) + ET(ta_i) \quad (11)$$

The objective function f_w is defined by (9) where f_1 and f_2 indicate minimizing the two objectives makespan and cost respectively. Equation (10) shows the makespan of the workflow that depends on the finish time FT of the exit task. The finish time of ith task is estimated as the sum of the start time and execution time of the same task in (11).

$$ST(ta_i)=\begin{cases}\max\left(AT(m_r)\right) & i=ta_{entry}\\ \max\left\{\left(AT(m_r)\right), \max\limits_{ta_j\in pred(ta_i)}\left(FT_j+CT_{ji}\right)\right\} & i\neq ta_{entry}\end{cases} \quad (12)$$

The start time ST given in (12) is calculated for an entry task ta_{entry} as the maximum available time of a virtual machine m_r for running the task. Otherwise the start time of all other nodes are calculated as follows:

i. Calculate the available time AT of a VM m_r for running a task.
ii. Sum the finish time FT and computation cost CT of the task ta_j in the predecessor set $pred(ta_i)$.

Then the maximum value of the results derived from steps (i) and (ii) is being chosen as the ST. The execution time ET of i^{th} task is calculated as the division of its file size FS by the computing capacity CC_r of the VM which is running the task given in (7).

D. Proposed EHEFT Heuristic

EHEFT is an extension of HEFT and thus can be divided into two phases: task prioritization phase and resource selection phase. Each task is assigned a priority using upward rank as described in HEFT in the primary phase of EHEFT, and is given by (5). Then, the workflow tasks are sorted in decreasing order of their ranks. Higher precedence is given to the task with higher rank.

The EHEFT heuristic integrates the mechanism of *crowding distance* computation into the resource selection phase of HEFT to select the tasks that have the same rank value. The standard formula for calculating crowding distance is given by

$$CD_i=\left(\frac{F_i-F_i^{\min}}{F_i^{\max}-F_i^{\min}}\right) \quad (13)$$

The core of the crowding distance is to find the Euclidian distance between the equal ranked tasks and is given by (13). The above formula is redefined to assign priority for the equal ranked tasks is given in (14).

$$CDC_i=\begin{cases}F_i & n=0\\ (2n+1)F_i & n=1\ or\ F_i^{\max}=F_i^{\min}\\ (n+1)\left(\frac{F_i-F_i^{\min}}{F_i^{\max}-F_i^{\min}}\right) & otherwise\end{cases} \quad (14)$$

The crowding distance is a criterion to select a task having the same rank. Here F_i is the execution time of a task ta_i. If a task ta_i has no successors, then the crowding distance constraint is its execution time. A task ta_i has 1 successor (n=1), then CDC_i is calculated by multiplying (2n+1) with its execution time. Otherwise CDC_i is calculated by multiplying (n+1) with standard crowding distance CD_i.

The proposed EHEFT algorithm terminates while all the tasks in line with their rank are scheduled. Though it seems to be a small change, it has a significant effect in reducing makespan and cost than HEFT. Our improved EHEFT makes clear that the improvement has a stronger capacity to converge to the Pareto front.

E. Pseudo code for the proposed algorithm

```
1. Using (6),compute the average execution time for every task .
2. Compute the average communication time between tasks and
   their successors for each task according to (8).
3.for each task $ta_i$ in the graph do
```

```
      if task $ta_i$ is the last task then
          the urank value of $ta_i$ = its average communication time
      else
          compute urank value for each task according to (5)
      end
    end
4. Arrange the tasks in a scheduling list Q by
   decreasing order of rank values of tasks.
5. while Q is not empty do
      if urank($ta_i$)=urank($ta_j$) then
          compute new priority value for selecting DAG path
          and execute according to (14)
      end
   end while
6. Calculate total execution cost using
      $f_2 = \sum_{i=1}^{p}(PCS_i * MP_i) + (PCBW_i * FS_i)$
7. Calculate makespan using (10)
```

IV. PERFORMANCE EVALUATION

Performance evaluation of workflow scheduling techniques in a large scaled cloud under different system and configuration is very difficult. Workflowsim is one of the best toolkits that can be used to model datacentres, service brokers, and virtual machines of a cloud for experimenting scheduling policies. The proposed heuristic was implemented in Workflowsim toolkit [15] which is an extension of Cloudsim [16].

A. Performance Metrics

The goals of the proposed heuristic are to minimize the makespan of tasks as well as the total execution cost. Makespan and cost are the two metrics used to evaluate the EHEFT.

Makespan: The maximum required time to finish the execution of all tasks. It is calculated as per (10).

Total cost: The total cost of task ta_i on resource m_r is calculated based on the objective function f_2.

B. Result Analysis

Scheduling all the tasks of a workflow on a given number of processors without changing the precedence constraints leads to an optimal workflow scheduling algorithm. The heuristic is evaluated on a common and real set of workflow applications. In this work, three types of real world workflows developed in Pegasus toolkit and provided by the Pegasus workflow management system are considered [17]. These scientific workflows are:

- **Montage**: Montage application is an I/O intensive astronomical application created by NASA/ IPAC used to create custom mosaics of the sky using input images.
- **CyberShake**: The CyberShake workflow is a data-intensive application used by the Southern California Earthquake Centre to generate probabilistic seismic hazard curves for a region.

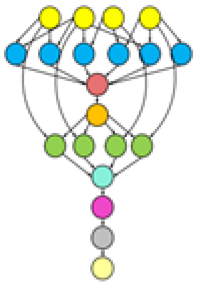
- **LIGO Inspiral**: LIGO Inspiral Analysis is a CPU intensive workflow used for gravitational physics.

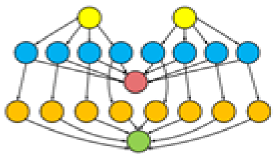
The coarse-grained structure of a small instance of each workflow utilized in our experiments is shown in Fig.2. For each workflow, tasks with the same shade belong to the similar kind. It can be seen that these three workflows have special structures, data and computational needs.

. These workflows are considered for evaluation with different categories given in Table II.

Table- II: Categories of Workflows

Size	**No.of Tasks**
Small	30
Medium	50
Large	100
Extra-large	1000


(a) Montage


(b) CyberShake

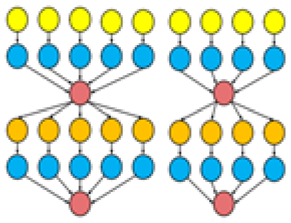

(c) Ligo Inspiral

Fig. 2. The scientific workflows

The hardware requirements as well as the configuration parameters used for the implementation of EHEFT in Workflowsim simulator are given in Table IV. Simulated datacentre (DC) host has 5 virtual machines (VMs) which are provided to users as resources. A datacentre (DC) is assumed to be having 1 CPU with a capacity of 1000 MIPS and 1000MB of available bandwidth. The costs for using memory, storage, bandwidth and processing cost are 0.05, 0.1, 0.1 and 3.0 units respectively in Table III.

Table -III: Configuration Parameters for running EHEFT in Workflowsim

No. of VM	5
Image Size	10000 MB
RAM	512 MB
MIPS	1000
Bandwidth(BW)	1000
PES Number (Number of CPU)	1

Processing cost for a resource	3.0 unit
The cost for using memory of a resource	0.05 unit
The cost for using storage of a resource	0.1 unit
The cost for using Bandwidth of a resource	0.1 unit

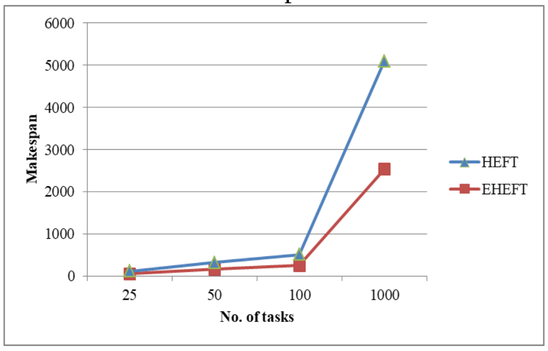
The EHEFT is examined using the three scientific workflows Montage, Cybeshake and Ligo Inspiral to validate its efficiency. EHEFT algorithm is implemented through Workflowsim and the performance measures are calculated for the Montage workflow for 25 tasks with its execution status are given in Table IV.

Table- IV: Simulation result of EHEFT for Montage 25 in Workflowsim

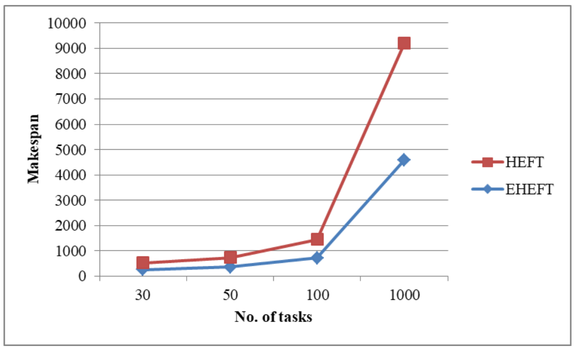
Cloudlet ID	STATUS	Data center ID	VM ID	Start Time	Finish Time	Depth	Cost
25	SUCCESS	2	0	0.1	0.16	0	2.33
2	SUCCESS	2	4	0.21	13.46	1	41.29
0	SUCCESS	2	3	0.21	13.49	1	41.38
3	SUCCESS	2	1	0.21	13.69	1	42.01
4	SUCCESS	2	2	0.21	13.87	1	42.55
1	SUCCESS	2	0	0.21	13.92	1	42.7
13	SUCCESS	2	1	13.99	24.2	2	32.73
8	SUCCESS	2	2	13.6	24.25	2	34.05
10	SUCCESS	2	3	13.99	24.34	2	32.35
6	SUCCESS	2	4	14.04	24.48	2	33.42
7	SUCCESS	2	0	14.04	24.76	2	34.26
11	SUCCESS	2	3	24.42	34.72	2	33.18
12	SUCCESS	2	1	24.37	34.78	2	33.48
5	SUCCESS	2	2	24.51	34.89	2	33.42
9	SUCCESS	2	4	24.65	34.91	2	33.07
14	SUCCESS	2	2	35.14	35.63	3	2.16
15	SUCCESS	2	2	35.86	37.04	4	4.26
16	SUCCESS	2	4	37.28	47.38	5	32.77
17	SUCCESS	2	3	37.28	47.63	5	33.52
20	SUCCESS	2	1	37.28	47.75	5	33.88
18	SUCCESS	2	0	37.28	47.83	5	34.11
19	SUCCESS	2	2	37.28	47.93	5	34.41
21	SUCCESS	2	2	48.22	49.34	6	8.27
22	SUCCESS	2	2	49.64	52.36	7	18.29
23	SUCCESS	2	2	52.67	56.2	8	20.88
24	SUCCESS	2	2	56.53	56.65	9	1.45
Makespan is : 56.65							
The total cost is : 735.7							

From the results given in Table IV, all the tasks are processed simultaneously. The makespan and the total cost required to run the workflow are also calculated.

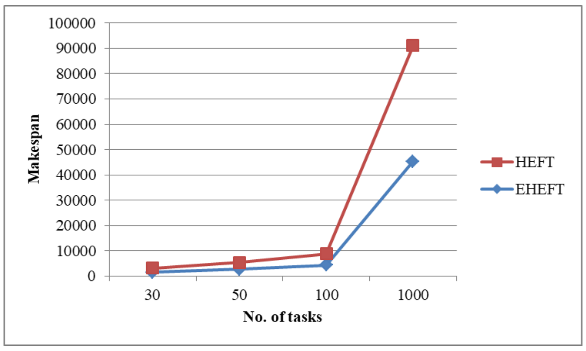
The results of EHEFT are compared with the HEFT algorithm for the workflows Montage, Cybershake and Ligo Inspiral with reference to makespan and cost. The comparison of makespan between the EHEFT and HEFT is presented in Fig. 3. In order to visualize this stark comparison, the results are plotted using a stacked line chart. The horizontal axis represents the workflow application considered for the experiments. The vertical axis gives the actual makespan taken by HEFT and EHEFT algorithm. The results show that EHEFT increases the efficiency of the VM in executing various tasks for all the three workflows and thus decreases the makespan.



(a) Montage workflows



b) Cybershake workflows



c) Ligo Inspiral workflows

Fig. 3 Simulation results of makespan for the workflows

The total cost required for the tasks per schedule when applied EHEFT and HEFT for the workflows Montage, Cybershake and Inspiral are compared and tabulated in Table V. The results indicate that the proposed heuristic decreases the total cost than that of HEFT.

Table -V: Simulation Results for the three Workflows

Data set	No.of nodes	Algorithm	Total cost	Makespan
Montage	25	EHEFT	735.7	56.65
		HEFT	736.25	56.98
	50	EHEFT	1626.7	160.28
		HEFT	1627.3	161.27
	100	EHEFT	3426.95	254.82

		HEFT	3427.78	256.69
	1000	EHEFT	36003.63	2536.2
		HEFT	36004.38	2556.19
Cybershake	25	EHEFT	19454.1	261.02
		HEFT	19454.65	262.39
	50	EHEFT	38315.76	362.9
		HEFT	38316.36	365
	100	EHEFT	76692.72	722.02
		HEFT	76693.42	726.95
	1000	EHEFT	127840.15	4588.78
		HEFT	127840.9	4623.1
Ligo Inspiral	30	EHEFT	19957.27	1556.15
		HEFT	19957.82	1564.03
	50	EHEFT	35467.19	2680.91
		HEFT	35467.79	2695.84
	100	EHEFT	63425.38	4361.8
		HEFT	63426.08	4390.54
	1000	EHEFT	686719.11	45394.5
		HEFT	686719.86	45716.5

This work can be applied to deal with real scientific workflow applications in cloud environment of distinct capacities at different costs. The outcomes of the experiments reveal that EHEFT gives improved results than HEFT. The comparison analysis of makespan and cost evidently depicts that the EHEFT workflow scheduling heuristic performs much better than HEFT.

Extensive simulations were conducted for obtaining the performance. Evaluation and comparison of EHEFT in terms of makespan and total cost with the selected heuristics DHEFT, MaxMin, MinMin and MCT were undertaken. The total cost and makespan results for the Montage workflow is depicted in Table VI.

Table -VI: Simulation results for Montage

Dataset	**Algorithm**	**Makespan**	**Total cost**
Montage_25	EHEFT	56.65	735.7
	DHEFT	57.1	736.23
	MINMIN	57.27	736.23
	MAXMIN	57.01	736.25
	MCT	57.1	736.53
Montage_50	EHEFT	160.28	1626.7
	DHEFT	142.41	1627.42
	MINMIN	132.59	1627.16
	MAXMIN	129.51	1627.96
	MCT	129.77	1627.44
Montage_100	EHEFT	254.82	3426.95
	DHEFT	266.79	3427.78
	MINMIN	269.97	3427.81
	MAXMIN	256.74	3429.09
	MCT	256.79	3427.53

The results of Montage application shows that the EHEFT heuristic reduces the makespan and total cost compared to the DHEFT, MinMin, MaxMin and MCT.

Table -VII: Simulation results for Cybershake

Dataset	Algorithm	Makespan	Total cost
CyberShake_30	EHEFT	261.02	19454.1
	DHEFT	318.55	19456.3
	MINMIN	285.73	19454.65
	MAXMIN	263.44	19454.66
	MCT	285.73	19454.64
CyberShake_50	EHEFT	362.9	38315.76
	DHEFT	499.81	38314.16
	MINMIN	389.96	38315.49
	MAXMIN	363.07	38318.36
	MCT	372.21	38316.28
CyberShake_100	EHEFT	722.02	76692.72
	DHEFT	776.98	76703.69
	MINMIN	859.15	76697.57
	MAXMIN	723.39	76694.85
	MCT	728.99	76698.49

The simulation results produced by EHEFT, DHEFT, MinMin, MaxMin and MCT heuristics towards makespan and total cost for Cybershake workflows with 30, 50 and 100 nodes are exhibited in Table VII. In case of makespan, EHEFT gains an improvement than the comparative algorithms for all of the Cybershake applications. Compared with DHEFT, MinMin, MaxMin and MCT, the total cost for EHEFT is statistically better in each case.

Table-VIII: Simulation results for Ligo Inspiral

Dataset	Algorithm	Makespan	Total cost
Ligo Inspiral_30	EHEFT	1556.15	19957.27
	DHEFT	1679.39	19957.45
	MINMIN	2008.82	19958.4
	MAXMIN	1657.98	19958.3
	MCT	1713.61	19958.39
Ligo Inspiral_50	EHEFT	2680.91	35467.19
	DHEFT	2834.04	35467.51
	MINMIN	3095.91	35468.5
	MAXMIN	2833.29	35467.97
	MCT	2715.77	35468.16
Ligo Inspiral_100	EHEFT	4361.8	63425.38
	DHEFT	4704.19	63427.11
	MINMIN	4598.11	63427.13
	MAXMIN	4403.09	63426.57
	MCT	4444.6	63427.22

The produced makespan and the total cost for executing the Ligo Inspiral application with nodes 30,50 and 100 using the EHEFT and comparative algorithms DHEFT, MinMin, MaxMin, and MCT is given in Table VIII. The obtained results show that the EHEFT algorithm optimizes the makespan with less in execution cost in comparison with the selected heuristics.

V. CONCLUSION

Scheduling of workflows is one of the important issues in cloud computing. To attain good system performance, an efficient workflow scheduling is essential. This paper focuses on the popular workflow scheduling heuristic HEFT, analysing and devising a new workflow scheduling algorithm EHEFT for cloud environment. Evaluation of the proposed algorithm is done with the Montage, Cybershake and Ligo Inspiral workflows having different sizes and different structures. The proposed algorithm is compared with HEFT, MaxMin, MinMin, MCT, DHEFT heuristics, under makespan and cost constraints. According to the experimental results, it is clearly evident that EHEFT is more efficient when compared with the other algorithms.

REFERENCES

1. Simsy Xavier, S.P. Jeno Lovesum, “Survey of Various Workflow Scheduling Algorithms in Cloud Environment”, International Journal of Scientific and Research Publications, Volume 3, Issue 2, ISSN: 2250-3153, Febrauary 2013.
2. K.Nithyanandakumari, S.Sivakumar, “Simulation of a Scheduling Strategy for Dependent Tasks in Cloud Computing”, International Journal of Computational and Applied Mathematics, Volume 12, Number 1, ISSN 1819-4966 © Research India Publications, 2017.
3. K. Nithyanandakumari, S. Sivakumar, “ A Study on DAG Model for Task Scheduling in Cloud Environment”, Proceedings of International Conference on Advanced Computing and Communication Systems *(ICACCS -2017),* IEEE ISBN No. 978-1-5090-4558-7,January 2017.
4. S.H.H Madni, M.S. Abdul Latiff, M.Abdullahi, S.M. Abdul hamid, M.J. Usman, “Performance Comparison of Heuristic Algorithms for Task Scheduling in IaaS Cloud Computing Environment”, PLoS ONE 12(5): e0176321, 2017.
5. B. Kruatrachue, T. Lewis, “Duplication Scheduling Heuristics (DSH): A New Precedence Task Scheduler for Parallel Processor Systems”, Technical Report, OR 97331, Oregon State University, Corvallis, 1987.
6. Kalka Dubey, Mohit Kumar, S.C. Sharma, “Modified HEFT Algorithm for Task Scheduling in Cloud Environment”,Procedia Computer Science Elsevier 125,pp. 725-732, 2018.
7. Rizos Sakellariou and Henan Zhao, “A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems,” 18th International Symposium on Parallel and Distributed Processing. IEEE, 2004.
8. J.J. Durillo, R.Prodan, “Multi Objective Workflow Scheduling in Amazon EC2”, Cluster Computing, Volume:2, Issue:17,pp. 169–189 ,© Springer Science-Business Media, , 2014.
9. Cui Lin, Shiyong Lu, “Scheduling Scientific Workflows Elastically for Cloud Computing”, 4th IEEE International Conference on Cloud Computing, 978-0-7695-4460-1/11, 2011.
10. W.Zheng,, R.Sakellariou,“Budget-deadline Constrained Workflow Planning for Admission Control”, Journal of Grid Computing,Volume:4, Issue:11, pp.633–651,2013.
11. K. Deb, “Multi-objective Optimization using Evolutionary Algorithms”, Wiley-Interscience Series in Systems and Optimization, Wiley, Chichester, 2001.
12. Xiang xiang, Chu Xinjie Yu, “Improved Crowding Distance for NSGA-II”, Neural and Evolutionary Computing, Cornell University, arXiv: 1811.12667, 2019.

13. R. Carlo, C. Raquel Prospero, Naval Jr., "An Effective Use of Crowding Distance in Multi-objective Particle Swarm Optimization", GECCO '05, Washington DC, ACM 1-59593-010-8/05/0006, June 25-29, 2005.
14. Samadi Yasser, Zobakh Mostapha, Tadonki Claude,"E-HEFT: Enhancement Heterogeneous Earliest Finish Time algorithm for Task Scheduling based on Load Balancing in Cloud Computing", Proceedings of HPCS 2018, Orléans, France. pp.601-609,2018.
15. W.Chen,E. Deelman," WorkflowSim: A Toolkit for Simulating Scientific Workflows in Distributed Environments", Proceedings of the 8th IEEE International Conference on E-Science (e-Science), Chicago, pp. 1–12, doi:10.1109/eScience.2012.6404430, 2012.
16. R..N.Calheiros, R. Ranjan, A.Beloglazov, C.A.F. de Rose, Raj Kumar Buyya, "CloudSim: A Toolkit for Modelling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", Software Practice and Experience, 41, pp.23–50, doi:10.1002/spe.995, 2011.
17. Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreirada Silva, Miron Livny,Kent Wenger, "Pegasus, a Workflow Management System for Science Automation", volume 46, Future Generation Computer Systems, Elsevier, ,pp.17-35.
18. S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. H. Su, K.Vahi, "Characterization of Scientific Workflows" ,Third Workshop on. IEEE, pp.1-10, 2008.