Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes

Tao Hai[1,2], Jincheng Zhou[1,3*], Dayang Jawawi[2], Dan Wang[3,4], Uzoma Oduah[5*], Cresantus Biamba[6*] and Sanjiv Kumar Jain[7]

Abstract

Cloud computing is an extremely important infrastructure used to perform tasks over processing units. Despite its numerous benefits, a cloud platform has several challenges preventing it from carrying out an efficient workflow submission. One of these is linked to task scheduling. An optimization problem related to this is the maximal determination of cloud computing scheduling criteria. Existing methods have been unable to find the quality of service (QoS) limits of users- like meeting the economic restrictions and reduction of the makespan. Of all these methods, the Heterogeneous Earliest Finish Time (HEFT) algorithm produces the maximum outcomes for scheduling tasks in a heterogeneous environment in a reduced time. Reviewed literature proves that HEFT is efficient in terms of execution time and quality of schedule. The HEFT algorithm makes use of average communication and computation costs as weights in the DAG. In some cases, however, the average cost of computation and selecting the first empty slot may not be enough for a good solution to be produced. In this paper, we propose different HEFT algorithm versions altered to produce improved results. In the first stage (rank generation), we execute several methodologies to calculate the ranks, and in the second stage, we alter how the empty slots are selected for the task scheduling. These alterations do not add any cost to the primary HEFT algorithm, and reduce the makespan of the virtual machines' workflow submissions. Our findings suggest that the altered versions of the HEFT algorithm have a better performance than the basic HEFT algorithm regarding decreased schedule length of the workflow problems.

Keywords HEFT Algorithm, Cloud Computing, Task Scheduling, NP-complete

*Correspondence:
Jincheng Zhou
guideaaa@126.com
Uzoma Oduah
uoduah@unilag.edu.ng
Cresantus Biamba
cresantus.biamba@hig.se
[1] School of Computer and Information, Qiannan Normal University for Nationalities, Duyun 558000, China
[2] Faculty of Computing, Universiti Teknologi Malaysia (UTM), 81310 UTM Skudai, Johor Bahru, Johor, Malaysia
[3] Key Laboratory of Complex Systems and Intelligent Optimization of Guizhou, Duyun 558000, China
[4] School of Mathematics and Statistics, Qiannan Normal University for Nationalities, Duyun 558000, China
[5] Department of Physics, Faculty of Science, University of Lagos, Lagos 100213, Nigeria
[6] School of Mathematics and Statistics, Department of Educational Sciences, Faculty of Education and Business Studies, University of Gävle, Gävle, Sweden
[7] Electrical Engineering Department, Medi-Caps University, Indore 452012, India

Introduction

Cloud computing works on a "pay for each use" system where clients access the cloud services without having full knowledge of the distribution policies and hosting specifics [1–3]. This provides global on-request access to a shared pool of assets such as storage space, computing servers, and web facilities for a reduced time to shop for enterprises and determine the logical findings [4]. Clients can access these assets steadily with no stress and no need to communicate with the facility provider [5, 6]. The aim of cloud infrastructure is to provide an easy-to-use workspace for dynamic applications.

The workspace can be obtained when various computer hardware are integrated with software package services. These facilities allow clients to transmit their submissions in cyberspace through the indication of their execution, accessibility, and Quality of Service (QoS) necessities [7]. As a result of the different configuration, deployment, and arrangement necessities of such submissions, the approaches for asset management and task scheduling becomes basic in the development of the efficiency and effectiveness of the cloud framework [8, 9]. In a distributed framework, all the jobs may be imagined as executing the various tasks in it. These tasks are classified into dependent and independent tasks. While independent tasks can be performed concurrently by several Virtual Machines (VMs), dependent tasks have to be planned through the fulfilment of their precedence relationships. This can be presented as a Directed Acyclic Graph (DAG) where the graph vertices or nodes represent tasks, and edges represent links between the tasks [10, 11]. It is compulsory to perform tasks with precedence restrictions in a scheduling order that decreases the schedule makespan. NP-Complete is the discovery of the maximal results for a task scheduling challenge [10].

Task scheduling issues can be classified into two primary classes: the deterministic and non-deterministic scheduling. The deterministic (compile-time) scheduling is sub-divided into the heuristics-based [12, 13] and Guided Random Search-Based (GRSB) [14–16]. Deterministic task scheduling is also referred to as static scheduling. The GRSB algorithms (Genetic Algorithms) cost more than heuristics-based scheduling algorithms because the algorithms need more iterations to generate an enhanced schedule. The heuristics-based algorithms on the other hand, provide approximate solutions in record time. They can be categorized as duplication-related [17, 18], clustering-based [19, 20], and list-based [21–23]. The duplication-based heuristics have higher time complexity, while clustering-based heuristics are suitable for homogeneous frameworks.

In this paper, we considered list-based heuristics because of their decreased duration and efficiency in delivering a shorter makespan. They work in two primary stages for task scheduling. In the first stage, calculation of rank is done for individual tasks, after that arranged in a descending order. In the second stage, we schedule the task with the highest rank value on the available machine. The Heterogeneous Earliest Finish Time (HEFT) procedure is the most popular among its counterparts for heterogeneous computing because of its high performance trade-off and low costs [24].

The following are the main contributions of this study:

- We design and propose three altered versions of the HEFT algorithm for rank calculation and processor selection, and to reduce the duration for the task scheduling.
- We lay out the challenge of task scheduling on heterogeneous machines and the cloud framework-related features for efficiently managing the specified tasks on the available VMs through the inclusion of the dependency restrictions among the tasks.
- We analyse and compare the proposed algorithms with the basic HEFT algorithm, the AVCT (Average Computation Cost) algorithm on arbitrarily created DAGs of real-world applications.

The novelty of the proposed method lies in the different methodologies in the two stages of the HEFT algorithm. In the first stage (rank generation), we execute several methodologies to calculate the ranks, and in the second stage, we alter how the empty slots are selected for the task scheduling. These alterations do not add any cost to the primary HEFT algorithm, and reduce the makespan of the virtual machines' workflow submissions. From the computational analyses and experiments we carried out, we observed the significant differences between the performance of the basic HEFT algorithm (AVCT approach) and our proposed altered versions MXCT (Maximum Computation Cost), MNCT (Minimum Computation Cost), and AVBS (Average Computation Cost and Best Empty Slot), regarding the schedule makespan that was produced. This implies that the scheme used affects the schedule length. We also observed that using the average value scheme for rank calculation and selection of the first empty slot is not always the best option. Our findings indicate that our proposed improved versions perform better than the basic HEFT algorithm regarding the decreased schedule length of the workflow problems running on the virtual machines.

The rest of this paper is organized as follows: Section 2 reviews the related literature. Section 3 briefly introduces multiprocessor task scheduling, and describes the problem model. Section 4 explores the HEFT algorithm and the proposed methodology. Section 5 discusses the

experimental results. Finally, Section 6 concludes the paper.

Literature review

The authors in [5] proposed a community-based cloud framework to manage emergencies. Its aim is to coordinate and oversee different organizations and combine large amounts of heterogeneous data in order to deploy logistics and personnel to search and rescue. The framework can also be utilized in the assessment of damage. In [6], to make clear the fundamentals of cloud computing, the authors explained the features of the areas which distinguish cloud computing from other research areas. They mainly compared cloud computing to grid computing and gave insights to the essentials of both concepts. The authors in [7] proposed a toolkit which allows the simulation and modelling of application provisioning and cloud computing systems. The aim was to achieve resource performance and application workload models under different user and system configurations. In [8], the authors provided a brief but comprehensive overview into speech bifurcation, both into series and single words with unrestricted speech, and presented a methodology which converts vocal signals into text. The authors in [9] proposed a game theoretic framework for the management of dynamic cloud services, including allocation of resources and assignment of tasks, with the aim of providing reliable cloud services. The proposed framework would assist cloud service providers in the management of their resources in a cloud computing environment.

In [10], the authors presented an algorithm for scheduling of tasks that makes use of the standard deviation of the estimated task execution time on the resources available in the computing environment. This approach considers the heterogeneity of the task and significantly reduces the execution time of a specific application. The authors in [11] proposed improved versions of algorithms specifically for heterogeneous systems used for compilation of time list scheduling where the priorities of the tasks are computed. In [14], the authors examined the dynamic scheduling of tasks in a multiprocessor system in order to obtain a viable solution making use of genetic algorithms integrated with popular heuristics. The experimental results showed that the genetic algorithm can used for task scheduling to meet deadlines. The authors in [15] designed a genetic evolution-based algorithm to find an optimal solution for task scheduling in a multiprocessor system in record time. In [16], the authors provided a comprehensive overview of genetic algorithms, its techniques, tools and research results which would allow the algorithms to be applied to real-world problems in different fields. The authors in [12] presented two novel algorithms for heterogeneous processors with the goal of attaining speedy scheduling time and high performance. The experimental results revealed that the proposed algorithms performed better than existing algorithms in terms of quality and cost of schedules.

In [13], the authors proposed an algorithm for scheduling tasks in a multicore processor system which significantly decreases the recovery time in case the system fails. The proposed algorithm is based on a check pointing method. The authors in [17] proposed a cutting-edge duplication-based algorithm to reduce the schedule makespan and delay of the task execution. The proposed algorithm schedules tasks with the lowest redundant duplications. In [18], the authors presented a list scheduling algorithm to consider the heterogeneity of communication and computation. They also proposed a novel approach for priority computation which considers the difference in performances in the target computing system making use of variance. The authors in [23] proposed a ranking algorithm based on the parent–child relationship and the priority assignment stage of the HEFT algorithm designed for task scheduling in a multiprocessor system. The proposed algorithm works on the keywords' density, the age of the webpage, and the amount of node successors.

The problem model

Multiprocessor task scheduling

Previously, various researchers have proposed several list scheduling procedures to resolve the task scheduling issue. The HEFT algorithm [12] estimates the tasks' ascendant rank values with the average communication and computation cost. The Standard Deviation Based Task Scheduling (SDBATS) [10] uses the standard deviation of transmission and computational expenses to approximate ascendant rank values. The Critical Path on a Processor (CPOP) [12] adds the descendant and ascendant rank values to create an important track and precedence column. During each stage of the DAG, the Performance Effective Task Scheduling (PETS) [25] includes the average computation cost, data transmission and reception cost to fix the tasks' rank values. The Duplication based Heterogeneous Earliest Finish time (HEFD) [18] uses task variance as a feature of heterogeneity to approximate the transmission and computation costs among tasks. Predict Earliest Finish Time (PEFT) [24] is created on the look-forward technique, and approximates the descendent tasks through the calculation of an Optimistic Cost value Table (OCT). The OCT is a 2D array whose columns and rows indicate the number of processors and tasks, respectively. Each element in the OCT (t_i,p_j) shows the optimum of the shortest ways of t_i offspring tasks to the leaving node, noting

that machine p_j is nominated for task t_i. All these algorithm calculations rely on standard deviation or the average of task weights on accessible machines. They do not include the framework heterogeneity. The most recent effort shows how standard deviation includes task and heterogeneity on existing machines. The various task scheduling algorithms and their limitations, tools, and parameters were analyzed in [26, 27].

We believe that the HEFT algorithm's efficiency can be improved by using three versions of the basic HEFT algorithm. This paper proposes two schemes of the first stage (rank calculation), and a different approach for selection of the empty slot. We examined the schedules makespan generated by each version and regarded the minimum length makespan as the result. Although it slightly increases the algorithm's costs, it is a trade-off between time complexity and performance. Our evaluation illustrates that the proposed versions produce high value schedules in terms of higher efficiency and decreased schedule length.

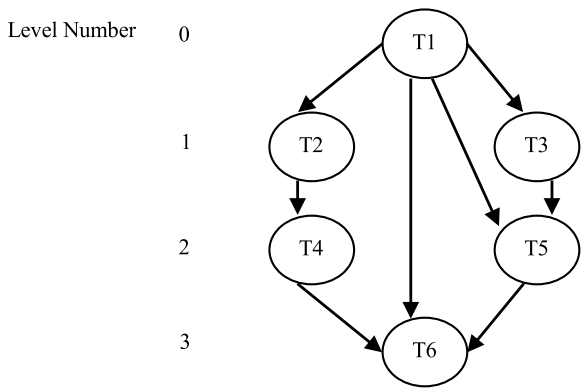

Fig. 1 A model DAG

The model and objective function

The model of the scheduling structure consists of a target computation architecture, a submission (application), and scheduling standards. A problem can be indicated as a (DAG) = G (T, E, R, C) (see Fig. 1), where $T = \{t_i, i = 0, 1, 2, ..., n-1\}$ is a set of n tasks [28–30]. Symbol E indicates a set of edges between tasks $E = \{e_{i,j}, i < j\}$, and $e_{i,j}$ represents the precedence limitations between two linked tasks. Taskst_i,$t_j \in$ T, which are connected to each other, signifying the precedence limitation of task t_j being dependent on task t_i for its operation. It illustrates that task t_i results will be applied as the input value for task t_j, and t_j cannot begin its implementation beforet_i. The task t_j is the heir of t_i and t_i is the predecessor of t_j. Here, R signifies a 2D matrix of size$v \times m$, and r_{ij} in R denotes the estimated operating time of v_i on j^{th} processor. A matrix $CmC(t \times t)$ represents the communication cost between any two tasks t_i andt_j. In the graph below, a task with no ancestor is referred to as an entry task, and a task that has no descendant is referred to as an exit (leaving) task.

A cloud framework consists of a set $VM = \{vm_i where i = 0, 1, 2, \ldots . m-1\}$ of m VMs that self-regulate and are linked over a high-rate network as illustrated in Fig. 2. The Data Transfer Frequency (DTF) may change because of the modified network bandwidth of cloud architecture. DTF can be written as an $m \times m$ two-dimensional array, and among any two VMs as$DTFm \times m$. The Probable Execution Cost (PEC) can be indicated by an extra 2D array $PECn \times m$ to carry out a task t_i on a VM vm_j, where $0 <= i <= n-1$ and $0 <= j <= m-1$. The PEC builds on the VM's speed of computation and can be different for each VM.

The communication cost between vm_x and vm_y depends on two aspects. The first is the processors' installed frequency on both sides of communications. The second is the frequency's correspondence cost value. We assume

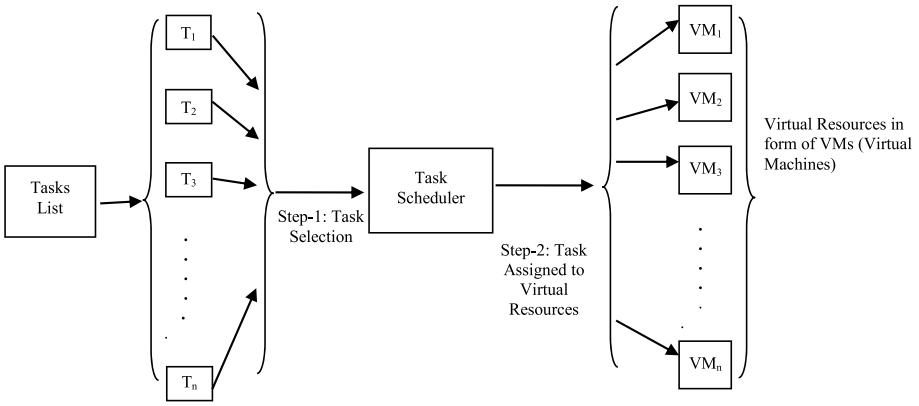

Fig. 2 Task scheduling in a cloud-based framework

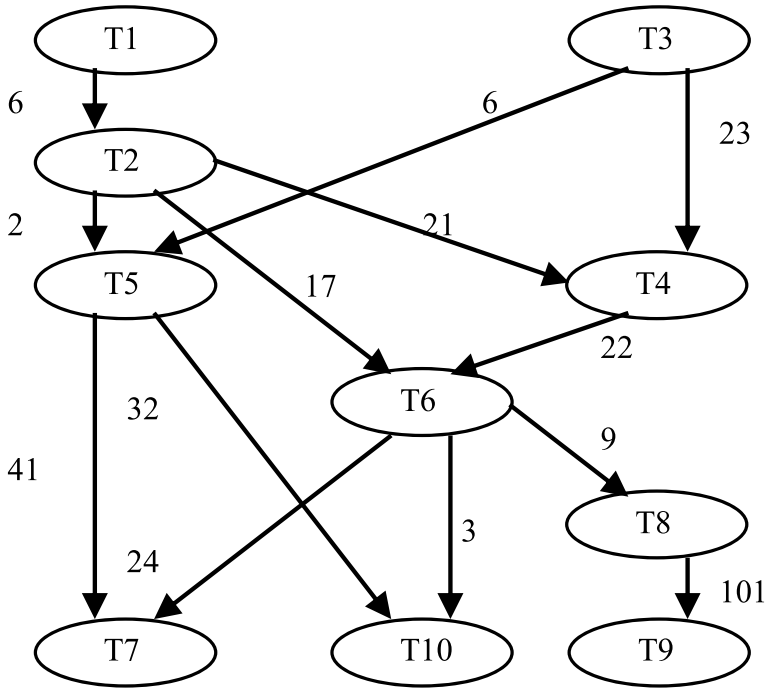

Fig. 3 Model task graph (using 10 tasks)

that each VM workstation can transmit to other workstations of a different VM with no conflict on the transmission channel. We also assume that tasks planned on the similar VM have no cost of communication among them.

The aim of the task arrangement challenge is to organize all the tasks of a given submission to machines for the application's completion time to reduce, fulfilling all the precedence limitations.

Methodology

Review of the HEFT algorithm

The HEFT algorithm is designed to schedule the DAG tasks into heterogeneous processors. The HEFT procedure has two basic stages: the rank generation and the processor selection stages. In the first stage, HEFT carries out a calculation of ranks for all the tasks and prioritizes them according to a descending order of their rank values. It is initiated by assigning weights to each DAG node and edge for rank calculation, based on the average communication and computation cost.

In the second stage, HEFT chooses the tasks based on their priority values and schedules each nominated task on the most suitable processor, which can decrease the schedule length of the task. HEFT also arranges the tasks in an empty slot between two previously planned tasks on a processor if the precedence constraints are observed. The HEFT algorithm looks for an empty slot on a processor until it finds one that can carry the computation cost of the chosen task.

HEFT procedure makes use of average communication and computation costs as DAG weights for calculation of ranks, and selection of processor. The first empty slot is always considered for the task scheduling. In some cases, however, the average cost of computation and selection of the first empty slot may not be a good solution. Consider the sample DAG in Fig. 3. The cost of probable execution of every task on three different VMs is shown in Table 1. The edges of the sample DAG are labelled with the average cost of communication.

Table 1 Probable Execution Cost (PEC) matrix

Task	VM1	VM2	VM3
T1	10	6	9
T2	10	23	23
T3	10	9	7
T4	18	17	18
T5	5	2	23
T6	7	5	16
T7	17	9	17
T8	40	16	19
T9	18	9	8
T10	26	10	24

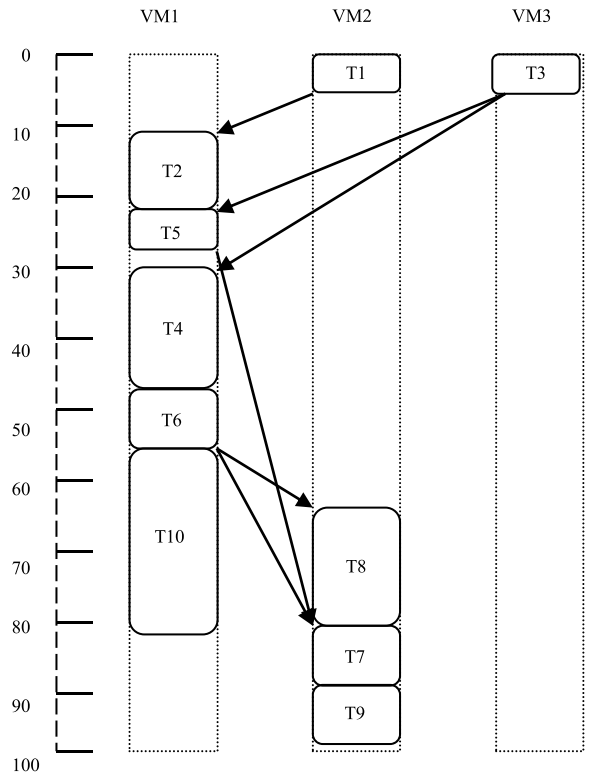

Fig.4 Scheduling of DAG with original HEFT algorithm. (Use of Average Computation Cost in Rank Calculation & Schedule length is 98)

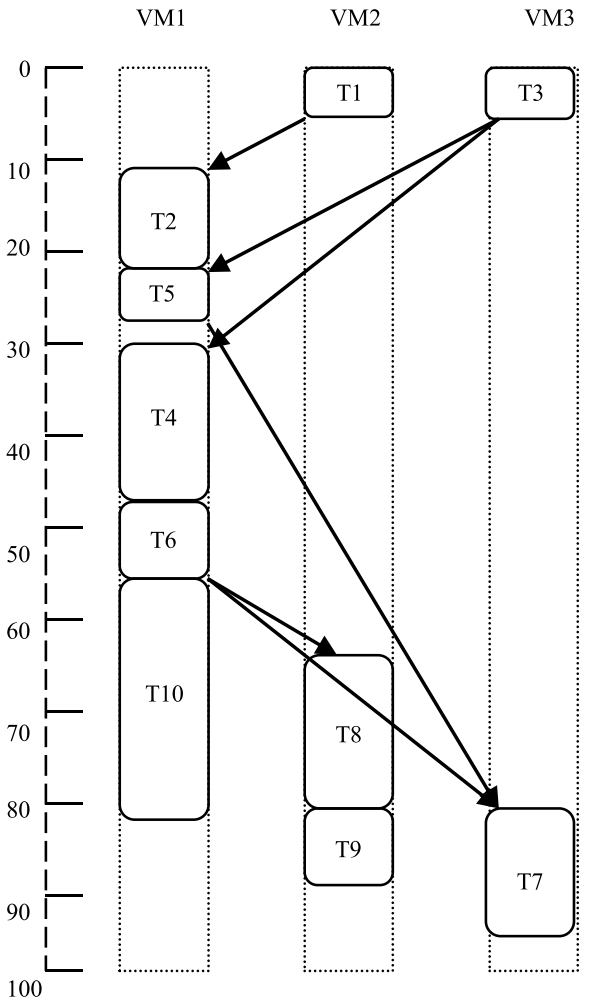

Fig.5 Scheduling of DAG with modified HEFT Algorithm. (Use Of maximum Computation Cost in Rank Calculation & Schedule length is 96)

In this example, if prioritization of tasks is done using the average cost of computation of all three VMs (as in the basic HEFT), then the scheduling order would be $T_1, T_2, T_3, T_4, T_6, T_8, T_5, T_{10}, T_7, T_9$, and the schedule length would be 98. Figure 4 illustrates this. Assume the assigning of priorities is done using the optimal value of the cost of computation over the three VMs on which a task may be executed. In such a situation, the task scheduling order would be $T_1, T_2, T_3, T_4, T_6, T_8, T_5, T_{10}, T_9, T_7$, and the schedule length would be 96. Figure 5 illustrates this. This is lesser than the schedule length obtained by the basic HEFT algorithm. Similarly, if priorities are assigned using the minimum value of the cost of computation over the three VMs on which a task may operate, and then the task scheduling length and order would be the same as found in the basic HEFT algorithm. This is illustrated in Fig. 6.

Conversely, if the processor selection stage is altered, the schedule length obtained may change. Here, the calculations of ranks are done using the average computation cost value. In the above example, if we choose an empty slot in which a task has the lowest finish time instead of the initial slot, the schedule length would go down to 89 as task T_5 is now scheduled on VM_2 (22 to 27, see Fig. 7) instead of VM_1 (24 to 26 in case of the basic HEFT, see Fig. 4). We find that the average cost of computation value for rank calculation and selection of the first empty slot for scheduling of tasks are not the best choices. The schedule lengths gotten may change.

The proposed methodology for cloud environment

We propose the altered versions of the basic HEFT algorithm to acquire improved results for task arrangement challenges in the cloud environment. Figure 8 illustrates this. In the first stage (rank generation), we execute a

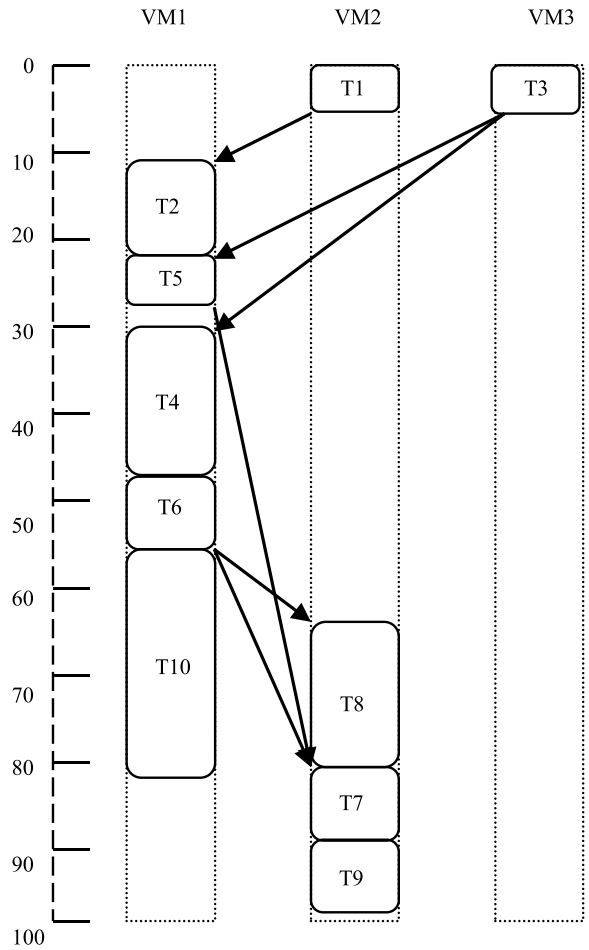

Fig.6 Scheduling of DAG with modified HEFT Algorithm. (Use of minimum computation cost in rank calculation & schedule length is 98)

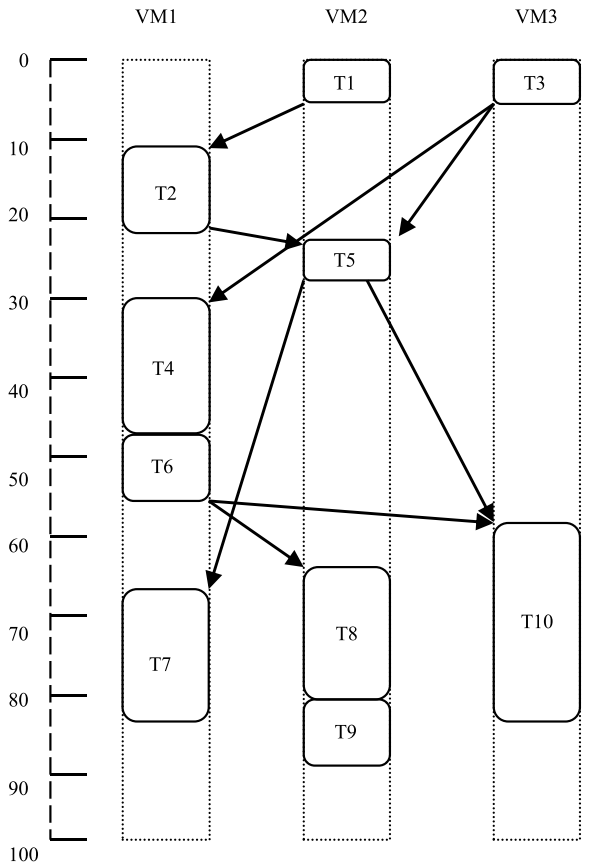

Fig.7 Scheduling of DAG with modified HEFT algorithm. (Use Of min schedule length idle slot in processor selection & Schedule length is 89)

distinct methodology, and in the second stage (resource selection), we alter the mode of selection of empty slots for task scheduling. These modifications do not incur any additional cost compared to the basic HEFT algorithm. The changes we propose in each stage are explained below.

Rank generation stage

In this stage, each task's precedence should be decided with the descendant or ascendant rank value. The formulas below calculate the task's upward rank:

If Task T_i is an exit task, the rank of task T_i is defined by the Rank Function:

$$Rank_{up}(T_i) = f(w_i^1, ..w_i^k, ..w_i^n) \tag{1}$$

Else,

$$Rank_{up}(T_i) = f(w_i^1, ..w_i^k, ..w_i^n) + \max_{\forall T_z \in suc(T_i)} \left(avg\left(comm_{i,z}\right) + Rank_{up}(T_z)\right) \tag{2}$$

where w_i^k is the computation amount of task T_i on resource k and $1 \leq k \leq n$, $suc(T_i)$ is the set of the direct successors of task T_i and $avg(comm_{i,z})$ is the average cost of communication between the tasks T_i and T_z. Here, f represents a function which could be the maximum, minimum or average value of the cost of computation. As the rank is calculated recursively from the exit node, it is referred to as the Upward Rank value.

HEFT ()

{

Step 1. *In DAG, Label the nodes and edges with the average values of computation cost and communication cost respectively.*

Step 2. *Calculate* $Rank_{up}(T_i) = f(w_i^1, .. w_i^k, .. w_i^n)$ / $Rank_{dw}(T_i) = 0$ *using equation (1) or (2)/ (3) or (4) for each task by passing through the DAG in upward/ downward direction, starting from the last task/ first task.*

Step 3. *Arrange the tasks in a scheduling queue in order of their decreasing* $Rank_{up}(T_i) = f(w_i^1, .. w_i^k, .. w_i^n)$ *or increasing* $Rank_{dw}(T_i) = 0$ *values.*

Step 4. ***While*** *there are unallocated tasks in the scheduling queue* ***do***

Step 5. *Pick the first task,* n_i*, from the scheduling queue.*

Step 6. *for each Virtual machine* p_k *in the machine set do*

Step 7. *Calculate Earliest Finish Time* (n_i, p_k) *value using the insertion based allocation policy.*

Step 8. *Allocate task* n_i *to the machine* p_j *that minimizes EFT value of task* n_i*.*

Step 9. ***End while.***

}

Fig. 8 The complete HEFT algorithm

If Task T_i is an entry task, the rank of Task T_i is defined by the Rank Function:

$$Rank_{dw}(T_i) = 0 \tag{3}$$

Else

$$Rank_{dw}(T_i) = \max_{\forall T_z \in pre(T_i)} (avg(comm_{z,i}) + Rank_{dw}(T_z) + f(w_i^1, .. w_i^k, .. w_i^n)) \tag{4}$$

where, $pre(T_i)$ is the group of direct predecessor of Task T_i. After calculating the each task rank, a task list is created through the arrangement of the tasks according to their descending order of $Rank_{up}$.

The Resource Selection Stage.

In this stage, we propose a novel approach to determine the empty slot for the selected task. Here, the search for an appropriate empty slot for a task on a resource starts when all the ancestors of Task T_i transmit the required input data to that resource. The search continues until an empty slot which can hold the cost of computation of Task T_i and in which the selected Task T_i has the lowest finish time is found. In Fig. 8 complete algorithm of *HEFT* is presented.

Results and discussion

In this paper, we created a system that complements the basic HEFT algorithm, and improves the processor selection and prioritization of task processes. Input fed into the system includes the cost of communication, the Probable Execution Cost Matrix, the DAG showing dependencies, and the number of VMs and tasks. To assess the performance of our algorithm, we generated varied scheduling problems and attempted to solve them with the altered versions of the HEFT algorithm, as well as the basic HEFT algorithm.

We designed an automated system to make scheduling issues of different sizes. This is done to prevent partiality

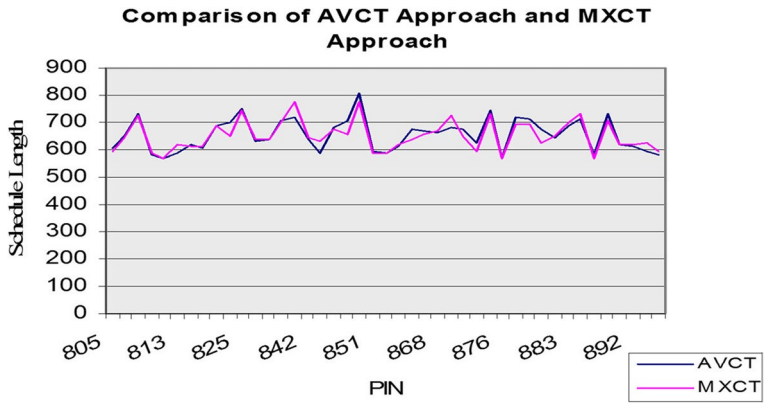

Fig. 9 Variations in schedule length obtained from the AVCT approach and MXCT approach

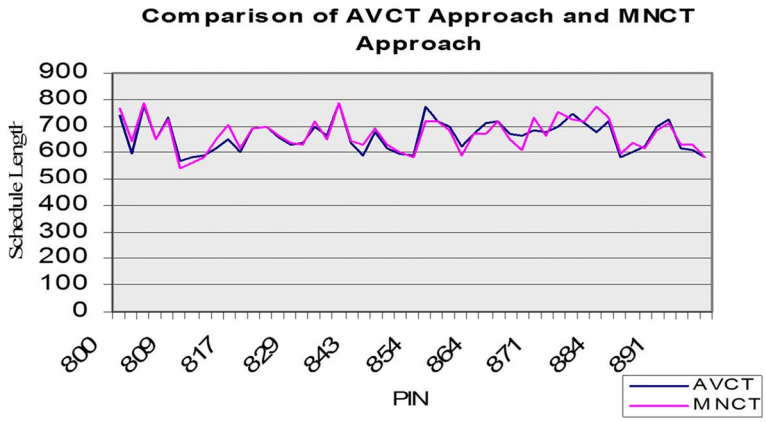

Fig. 10 Variations in schedule length obtained from the AVCT approach and MNCT approach

Table 2 Comparison of AVCT, MXCT and MNCT algorithms

Problem Percentage	Approaches	Outcomes	
33%	AVCT	Equal Schedule Length	
	MXCT		
	MNCT		
67%	MXCT Vs AVCT	36%	Equal Schedule Length
		39%	MXCT Gives better Schedule Length
		25%	MXCT Gives worse Schedule Length
	MNCT Vs AVCT	27%	Equal Schedule Length
		40%	MNCT Gives better Schedule Length
		33%	MNCT Gives worse Schedule Length

when offering values of different parameters. Our system assigns the parameters to random values in suitable ranges. We created problems for our experiments with the following features:

- Size of the problem (the number of tasks) which ranges from 50 to 80 with an interval of 5.
- The number of each task's successor with the exception of the exit task which is an arbitrary number that ranges from 0 to 10.

Table 3 Comparison of the AVCT and AVBS algorithms

Problem Size	Number of Resources	AVCT algorithm (Average of Hundred problems)	AVBS Algorithm (Average of hundred problems)
50	4	425.50	425.14
55	4	457.34	457.32
60	4	491.12	490.94
65	4	548.24	548.26
70	4	574.66	574.18
75	5	623.21	622.02
80	5	665.97	665.61

- The task implementation time, this is an arbitrary number which ranges from 1 to 20.
- The task communication time, this is an arbitrary number between 1 and 50.
- The amount of VMs is considered as either 4 or 5.
- In all the DAGs, the tasks' ranks are calculated using the upward rank calculation formula.

Investigation on the rank generation stage

To calculate a particular value for the method f in Rank Function, we use three techniques:

(i) AVCT (Average Computation cost) approach: This returns the average task cost of computation over every VM. It is used in the basic HEFT algorithm.
(ii) MXCT (Maximum computation cost) approach: This returns a maximum task cost of computation over every VM.
(iii) MNCT (Minimum computation cost) approach: This returns a minimum task cost of computation over every VM.

In all the approaches, the first empty slot which can hold the task cost of computation is studied (as in the basic HEFT). We operate the basic HEFT algorithm (AVCT approach) and our proposed approaches (MNCT and MXCT approach) on 100 different problems with different Problem Identification Numbers (PIN) 800 to 899 for problem size 80. The experimental results reveal that for 33% of problems, all the algorithms have equal length of schedule. For the remaining 67%, the MXCT approach provides equal length of schedule in 36%, better length of schedule in 39%, and worse length of schedule in 25% of the cases compared to the AVCT approach. The differences in the schedule lengths gotten from the AVCT and MXCT algorithms are illustrated in Fig. 9.

Similarly, for the remaining 67% of the problems, the MNCT approach provides equal length of schedule lengths in 27%, better length of schedule in 40%, and worse schedule lengths in 33% of the cases in comparison to the AVCT approach. The differences in the schedule lengths gotten from the AVCT and MNCT algorithms are illustrated in Fig. 10. Table 2 compares the three approaches.

From our analysis, we observed that there are noteworthy differences between the basic HEFT algorithm's performance (AVCT approach), and the altered versions (MNCT and MXCT). We also observed that using an average value scheme for rank calculation is not always the best choice.

Investigation on the resource selection stage

We used two approaches to select an empty slot for task scheduling:

(i) AVCT approach (the basic HEFT algorithm-average cost of computation and the first empty slot).
(ii) AVBS (average computation cost and best empty slot) approach: This approach uses the average cost of computation to calculate the ranks and selects an empty slot where the selected task has the lowest finish time.

We took 100 sample problems of each size ranging from 50 to 80 with an interval of 5. Table 3 shows how we ran both algorithms on sample problems, and how the results were analysed. We observed that that AVBS algorithm has a better performance than the AVCT algorithm. The results from the experiment shows that the AVBS algorithm has a reduced average length of schedule in 86% problem sets and slightly greater average schedule length in 14% problem sets in comparison to the AVCT algorithm.

Conclusion

In cloud computing, scalable resources are offered as services to clients through the Internet. Thus, a cloud service provider has more clients to attend to in the cloud computing architecture. As a result of this, task

scheduling is one of the biggest challenges in establishing a functional and efficient cloud computing environment. In this paper, we proposed different versions of the heuristic-based algorithm Heterogeneous Earliest Finish Time (HEFT) which carries out task scheduling and allocates resources in the cloud computing environment. On comparison of our proposed approach to other frameworks in terms of schedule length, we discovered that our approach performs better. We observed that the original HEFT algorithm 's efficiency can be enhanced by choosing the best result from each approach 's schedules. Although this may lead to the algorithm having a higher cost, it is a trade-off between cost and performance. We may further consider the Nature-inspired optimization algorithm-based scheduling for more effective task scheduling in the cloud environment. The existing work can be extended for dynamic task scheduling in the future.

Acknowledgements
This work was supported by UTM Research Fellow (No.00P27), the National Natural Science Foundation of China (No.61862051), the Science and Technology Foundation of Guizhou Province (No.[2019]1299, No.ZK[2022]449), the Top-notch Talent Program of Guizhou province (No.KY[2018]080), the Natural Science Foundation of Education of Guizhou province (No.[2019]203) and the Funds of Qiannan Normal University for Nationalities (No. qnsy2019rc09). The Educational Department of Guizhou under Grant NO. KY[2019]067.

Authors' contributions
Conceptualization by Tao Hai, Dan Wang; Methodology by Dayang Jawawi; Software by Jincheng Zhou; formal analysis by Dawang Jawani and Uzoma Oduah investigation by Tao Hai and Dan Wang; Resources and data collection by Jincheng Zhou, Cresantus Biamba; Writing by: Dan Wang, Sanjiv Kumar Jain and Tao Hai; Validation by: Uzoma Oduah, Sanjiv Kumar Jain and Jincheng Zhou; Funding Acquisition by Cresantus Biamba. The author(s) read and approved the final manuscript.

Funding
The project was supported by the Department of Culture Studies, Religious Studies and Educational Sciences, University of Gävle, Gävle, Sweden.

Availability of data and materials
The supporting data can be provided on request.

Declarations

Ethics approval and consent to participate
The research has consent for Ethical Approval and Consent to participate.

Consent for publication
Consent has been granted by all authors and there is no conflict.

Competing interests
There are no competing interests.

Received: 30 August 2022 Accepted: 18 September 2022
Published online: 27 January 2023

References

1. Zomaya AY (1996) Parallel and Distributed Computing Handbook. McGraw-Hill, New York
2. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Comp Syst 25(6):599–616
3. N. Tziritas, S. U. Khan, C.-Z. Xu, and J. Hong, (2012)"An optimal fully distributed algorithm to minimize the resource consumption of cloud applications," CoRR, vol. abs/1206.6207, .
4. Mr. Prince Gupta, Dr. Rajeev Sharma, Dr. Sachi Gupta, (2021) "Resource Management, Issues, Challenges and Future Directions in Fog Computing: A Comprehensive Survey", Published in Design Engineering, ISSN: 0011–9342, , Issue: 7, Pages: 14580–14593.
5. J. Li, Q. Li, S. Khan, and N. Ghani, (2011) "Community-based cloud for emergency management," in Proc. SoSE, .
6. Hashemi SM, Bardsiri AK (2012) Cloud computing vs. grid computing. ARPN J Syst Softw 2:188–194
7. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) Cloudsim: A toolkit for modelling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exper 41(1):23–50
8. Gaurav Agarwal, Dr. Vikas Maheshkar, Dr. Sushila Maheshkar, Dr. Sachi Gupta, (2018) "Vocal Mood Recognition: Text Dependent Sequential and Parallel Approach", Published in International Conference on Signals, Machines and Automation (SIGMA'18), February 23–25, .
9. Shin KS, Park M-J, Jung J-Y (2014) Dynamic task assignment and resource management in cloud services by using bargaining solution. Concurr Comput Pract Exp 26(7):1432–1452
10. E. U. Munir, S. Mohsin, A. Hussain, M.W. Nisar, and S. Ali, (2013) "SDBATS: A novel algorithm for task scheduling in heterogeneous computing systems." in Proc. IEEE IPDPS Workshops (IPDPSW).
11. Andrei Radulescu and Arjan J. C. van Gemund. (2000) Fast and effective task scheduling in Heterogeneous system. Proceedings of the 9th Heterogeneous Computing Workshop.
12. Topcuouglu H, Hariri S, Wu M-Y (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distributed Syst 13(3):260–274
13. S. Gotoda, M. Ito, and N. Shibata, (2012) "Task scheduling algorithm for multi-core processor system for minimizing recovery time in case of single node fault." in Proc. IEEE CCGRID, pp. 260–267.
14. Sachi Gupta, Vikas Mittal and Gaurav Agarwal, "Task Scheduling in Multiprocessor System Using Genetic Algorithm", in Proceedings of 2nd International Conference on Machine Learning and Computing (ICMLC-2010), ISBN: 9780769539775.
15. Sachi Gupta, Gaurav Agarwal and Vikas Mittal, (2013) "An Efficient and robust Genetic Algorithm for Multiprocessor Scheduling", in International Journal of Computer Theory and Engineering (IJCTE), Vol. 5, No.2, ISSN: 1793–8201.
16. Goldberg DC (1989) Genetic Algorithms in Search. Wesley publishing, Optimization and Machine Learning. Add
17. Jing Mei KL, Li K (2014) A resource-aware scheduling algorithm with reduced task duplication on heterogeneous computing systems. J Supercomputer 68(3):1347–1377
18. Tang X, Li K, Liao G, Li R (2010) List scheduling with duplication for heterogeneous computing systems. J Parallel Distributed Computing 70(4):323–329
19. Cirou B, Jeannot E (2001) Triplet. A clustering scheduling algorithm for heterogeneous systems. In proceedings of International Conference on Parallel Processing Workshop
20. Fiore U, Palmieri F, Castiglione A, De Santis A (2014) A cluster-based data-centric model for network-aware task scheduling in distributed systems. Int J Parallel Program 42(5):755–775
21. Gaurav Agarwal, Sachi Gupta, Praful Saxena and Saurabh Mukherjee, "Web Graph Based Ranking Algorithm for Search Engines", in Proceedings of International Conference on Network Communication and Computer (ICNCC-2011), ISBN: 9781424495504.
22. Sachi Gupta, Saurabh Mukherjee and Gaurav Agarwal, "List Scheduling Heuristic: Efficient Prioritization and Processor Selection Schemes for Heft Algorithm", in Proceedings of International Conference on Industrial Applications of Soft Computing Techniques (IIASCT-2011), ISBN: 9789381361221.
23. Gaurav Agarwal, Sachi Gupta and Saurabh Mukherjee, "Web Graph Based Search by Using Density of Keywords and Age Factor", in Proceedings of

International Conference on Computer Science and Information Technology (ICCSIT-2012), ISBN: 9789381693766.
24. Arabnejad H, Barbosa J (2013) List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table. Parallel and Distributed Systems, IEEE Transactions on 25(3):682–694
25. E. Ilavarasan, P. Thambidurai, and R. Mahilmannan, (2005) "Performance effective task scheduling algorithm for heterogeneous computing system." in Proc. ISPDC. IEEE Computer Society pp. 28–38.
26. Fard HM, Prodan R, Fahringer T (2014) Multi-objective list scheduling of workflow applications in distributed computing infrastructures. J Parallel Distributed Comp 74(3):2152–2165
27. Fard HM, Prodan R, Barrionuevo JJD, Fahringer T (2012) "A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments," 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), Ottawa, ON, Canada, pp. 300-309. https://doi.org/10.1109/CCGrid.2012.114
28. Dai Y, Zhang X (2014) A Synthesized Heuristic Task Scheduling Algorithm. Scientific World J 2014:9
29. Ali S, Siegel HJ, Maheswaran M, Hensgen DA, Ali S (2000) "Task execution time modelling for heterogeneous computing systems." in Heterogeneous Computing Workshop. pp 185–199
30. Agarwal G, Om H (2021) Parallel training models of deep belief network using MapReduce for the classifications of emotions. Int J Syst Assur Eng Manag. https://doi.org/10.1007/s13198-021-01394-3

Publisher's Note