Dynamic Systems and Applications 30 (2021) No. 10, 1653 - 1668

TIME OPTIMIZATION IN CLOUD COMPUTING WITH THE HETEROGENEOUS EARLIEST FINISH TIME ALGORITHM

Lokesh Sivanandam[a*], Sakthivel Periyasamy[b]And Uma Maheswari Oorkavalan[b]

a – Ramanujan Computing Centre, Anna University, Chennai – 600 025, India.

b - Department of Electronics and Communications Engineering, Anna University, Chennai-600 025, India.

* Corresponding author.
E-mail address: lokesh.s.phd@gmail.com

ABSTRACT. One of the most widely used platforms for executing works as processing elements via virtual machines is cloud computing. However, in order to use workflow apps effectively, a number of challenges must be solved. One of the most important issues in cloud computing is time optimization. The best scheduling of cloud computing works is an NP optimization problem for which several techniques have been presented. The Heterogeneous Early Finish Time (HEFT) technique is proposed in this paper to achieve time optimization across virtual machines, while attempting to reduce the completion time of a specific workflow application under a user-specified financial limit. To test its performance, the proposed optimization technique is compared with existing timing algorithms with respect to time optimization and performance. The task completion and runtime of the proposed algorithm is reduced, when compared to the existing algorithms.

Keywords: Cloud Computing, Time optimization, Heterogeneous Earliest Finish Time (HEFT), Task Scheduling

1. INTRODUCTION

The cloud computing model emerged with the growth of the internet and services that it provides to its users. The cloud model is built on distributed computing and comprises a collection of distinct VCs that are interrelated to generate computing resources and can be dynamic in nature. In order to increase resource utilization and earnings from resources by enhancing their economic efficiency, unoccupied resources must be utilized globally; the cloud model is best suited for this purpose. The cloud computing model's primary goal is to allow users to share resources and data. It is a platform that provides its users with services and apps. Cloud Computing provides three types of service software (SaaS), service platform (PaaS) and service infrastructure (IaaS) Cloud computing. These services are accessible to customers on a pay per usage basis, in which common computers, servers, data storage, application and networking resources are shared. The user is given with a software licensed service in SaaS on the basis of the subscription of services. These services can be accessed via the web browser from any machine. With PaaS users can use cloud services and then deploy their services on their own machines to create their own services. Customers can use the internet in IaaS organizational infrastructure. For the use of the infrastructure, the customer does not need to understand its internal architecture. The entire

Received JUL 12, 2021
www.dynamicpublishers.com;
www.dynamicpublishers.org;
ISSN1056-2176(Print); ISSN 2693-5295 (online)

https://doi.org/10.46719/dsa202130.10.08

infrastructure used by customers for business needs is used instead of being purchased as a basic rent when necessary and when customers have ceased to pay for services when the infrastructure requirements are no longer met. The number of cloud users has increased in the last year, so that the number of tasks must be managed in a proposal for this task planning.

The main objective of this work is to propose a scheduling technique that takes into consideration the variety and heterogeneity of virtual machines in a cloud computing cluster to lower the overall execution time of a process. As part of the same solution, it also considers data distribution and restrictions, i.e., the HEFT Algorithm schedules tasks and data transfers at the same time. As a result, the scheduler decides how works and data are spread among virtual machines, reducing overall on total execution time and data transfers. The major contribution of this work is to manage the time utilization of cloud computing platform. The CloudSim tool is then used to evaluate our scheduling strategy. A comparison study of certain state-of-the-art algorithms is performed to evaluate the performance of the enhancement algorithm.

2. LITRATURE SURVEY

Downward Cost Optimization algorithm (DCO) works by assigning the application's deadline constraint to each task's deadline constraint and meeting the task's deadline constraint from the bottom up. Furthermore, [1] In terms of cost minimization for parallel applications under a range of situations, proposed (Downward-upward cost optimization) DUCO surpasses existing approaches. This approach can provide a theoretical basis for QoSaware scheduling applications in heterogeneous cloud environments. SPO is a MapReduce work scheduling system for heterogeneous environments that is performance and security conscious. SPO is a two-level optimization method based on the HEFT algorithm that reduces total Map size. Reduce task execution time while maintaining security. Furthermore, numerous experiments utilizing synthetic real-world applications show that SPO is more effective and practical than Hadoop-stock. [3] Parallel computing was invented to solve the problems of serial computing. Task scheduling is a demanding and challenging problem in parallel computing. The DAG Model can be used to schedule multiple works or tasks in a simple and efficient manner. The purpose of scheduling is to reduce the time it takes for a parallel application to complete or finish by properly allocating works to the processors. The paper also discusses HEFT, a prominent DAG-based method.

A new task scheduling technique based on the minimal value characteristic was presented in [4]. The suggested approach was tested on two different DAG models, including the DAG1 model, which contains ten works and 15 dependent edges. This model was tested on two cloud servers with three virtual machines, and the proposed approach yielded 64 units of scheduling length for this model. In comparison to the HEFT algorithm, which has a scheduling length of 73 units, the proposed technique has a shorter scheduling length. [5] A DRHEFT strategy which begins with a listing algorithm which provides scheduling solutions then tries to locate the best SER-LTR deal-off solutions and identifies the best SER-LTR trade-overs. The DRHEFT suggested can simultaneously improve SER and LTR at cheap investment. [6] The proposed technique is tested on DAGs with 10 and 15 works, yielding 63 and 142 units, respectively, which are lower than the heuristic algorithm HEFT. This method might be extended by employing soft computing methods like GA and PSO to lower the makespan and other metrics, and it could be tested on a larger number of DAGs.[7] A DVFS technique-based energy-aware approach for fog-computing is presented. Power savings in the fog are obtained by using DVFS-fabricated processors that can be calculated at lower voltage and frequency during slack times. Furthermore, the evolving hybrid IWO-CA method is employed for task ordering without violating the previous constraints. In computing applications with predefined deadlines, the results of the experimental section show noticeable savings in energy.

Offer a First of maximum loss scheduling algorithm (FOML) that reduces total cluster energy consumption and average time. We then constructed the integrated CloudSim, Cloud Task Platform and Generator to check the effect of the FOML algorithm in this article. [8] The FOML algorithm cuts the average cluster completion time considerably compared to Min–Min, Max–Min, Suffrage and E-HEFT and hence reduces total energy use and user expenses. [9] CMSWC has developed a list of multi-target scheduling algorithms to address cost minimization and cover processes in IaaS clouds. Real-world workflow applications are used to test the performance of this method in terms of producing span-cost transactions. When compared to comparable methodologies, CMSWC consistently provides solutions with superior pricing and coverage for all processes studied. Finally, a variety of SSE strategies support the solution to the genuine Pareto front.[10] The heuristic HEFT schedule has been seeded as one chromosome of the starting population to get an optimum solution with the fewest number of iterations. To achieve optimal resource performance, the DCHG-TS additionally employs a load balance process. The

comparison of old and new algorithms yielded the following findings. DCHG-TS is superior than other calendar algorithms since it reduces both the cost and the duration of the schedule.[11] Task scheduling in the cloud environment is an important and prominent challenge. The performance of the cloud environment depends on algorithms to balance the load, which assign the appropriate VMs. The algorithm was simulated in a simulation environment for a cloud analyst that is a Cloudsim extension with enabled GUI. For a variety of DC and UB's, the results are compared. WWA is 2.5% better than PSO, 4.5% better than ACO, and 6.9% better than GA.

PSO using the Earliest Finish Time algorithm PEFT Predict. This work was designed to achieve an optimized PSO schedule using PEFT population that reduced costs and time. [12] In order to evaluate the performance of the proposed technique two parameters, time and costs were used. Experimental results show that the technical approach proposed takes less time and costs than current state-of-the-art and outperformed techniques.[13] Cloud's hybrid approach for scheduling scientific workflows is proposed for the deadline and budget constraints. As a solution to this problem, the suggested method incorporates the HEFT solution into the initial population to obtain optimum execution time and minimal execution costs. We expressed the many goals as several QoS functions, including time and money. [14] A modified electro-search strategy with genetic operators is proposed for efficient work allocation. The algorithm is then implemented in Cloudsim and compared to various contemporary algorithms such as HPSOGA, ES, GA, and ACO in terms of make span time, cost, and response time. When compared to the other algorithms considered, the suggested approach has a shorter make span, lower cost, and faster response time. The performance of HESGA, HPSOGA, ES, GA, and ACO was evaluated using a benchmark problem. HESGA outperforms current methods.[15] virtual machine (VM) utilization and complete time minimization for parallel task sets while meeting end-to-end deadline restrictions. The DRank and maxslack settings govern the order in which tasks in the workflow application are executed. The results of the experiments show that the suggested algorithm, ERPD, is comparable to three other well-known algorithms. Random workflow apps and real-world workflow applications, such as Montage, Cybershake, Epigenomics, and Genome, were utilized as benchmarks in the experiment.

For real-time provisioning in cloud-based industrial applications, a security-conscious dynamic planning technique is provided. To begin, [16] presented a three-level safety model and a heterogeneous two-level cloud architecture to handle the problem of configuration security in

the mobile cloud environment. A dynamic planning technique based on a mobile dynamic workflow scheme is also provided for real-time optimization in order to deal with dynamism in mobile clouds. [17] A new strategy for lowering average cloud-based data center energy asymptotically without losing computer speed or stability has been developed. As a result, when the cloud data center is lightly loaded, the suggested approach, which has been tested through simulations, can greatly reduce cloud service providers' energy consumption and transform power utilization into processing capacity when the platform is substantially loaded.

EMVO has been enhanced (Enhanced Multi-Version Optimizer). In a cloud environment, the EMVO was employed to optimize task scheduling. [18] The EMVO was used to map a number of activities to a desired number of VMs in order to save time, maximize output, and improve resource consumption. The MVO and PSO were both improved as a result of better resource use. [19] The novel BMDA approach has really been evaluated utilizing MATLAB software and fundamental metaheuristic algorithms such as the Dragonfly Algorithm (DA), the PSO, the MHS and whale optimization algorithm (bat) (WOA). [20] In the creation of a rim-cloud computing network a heuristic dynamic task processing technology is applied. Each edge node collects information from the surrounding nodes, updates the workflow dynamically and choose the optimal edge node to work on, and uploads applications services to the edge layer of the network.

3. PROPOSED METHOD

The HEFT algorithm is a DAG programming algorithm that can be used in a specific number of scenarios. The HEFT algorithm is divided into two phases: Set task priorities and pick tasks based on them in the task priority phase. The heterogeneous first-time (e-HEFT) approach is proposed in this article for achieving time optimization on virtual machines while attempting to reduce the completion time of a workflow application.

3.1 HEFT Algorithm (Heterogeneous Earliest Finish Time)

The HEFT algorithm requires a limited number of processors to programmers. The algorithm first establishes a priority task list and then performs optimal local allocation decisions for each task based on the projected completing time. Efficient planning is aimed at mapping the tasks to key processors and ensuring that the task needs are met and the minimum planning length defined. The HEFT algorithm offers a solution to the problem of DAG scheduling in

heterogeneous systems based on a strong efficiency, low working time and a capacity to supply steady performance in a wide array of visual structures. The implementation of techniques, which are all static ways to mapping and take on static conditions for a given time period, restricted the HEFT method.

As Figure 1 shown by a direct acyclic graph, the HEFT technique is used to distribute an application's work over a small number of heterogenic machines. Programmers are not included in this group. In HEFT, a variety of weight calculating algorithms are investigated. The average calculation and communication are used to determine the weight of each node and edge in the HEFT graph. The graph is then crossed up, and each node is given a grade value. The jobs are then placed on the machine in order of their rank value at the time they are completed.

3.1.1 Phase of Task Prioritization

Each schedule must have priority with the critical path length from task to exit task (i.e., the longest path) in the phase of the HEFT approach (Steps 1-5 in HEFT Algorithm). The task list is created by arranging the schedule in descending order of critical path length. The topological order of tasks in this length of directed acyclic network is a linear order of tasks that retains the primary criteria of the graph. For priority jobs, the Task Priority Strategy has been updated. During the processor selection step, tasks are assigned to the least HEFT processor to be executed according on the priority of the task scheduling order.

HEFT ALGORITHM

1. Allow the computation costs of works and the communication costs of edges to be averaged
2. Calculate the rank (u) of all tasks by traversing the graph upward, beginning with the exit task.
3. Sort tasks in a scheduling list in descending order of rank (u) values.
4. Do the following when there are unplanned works on the list:
5. To schedule, select the first task in the list, $\boldsymbol{ni}$.
6. Perform the following for each processor Pk in the processor-set ($\boldsymbol{Pk\ E\ P}$):
7. Compute Heterogeneous Earliest Execution Finish Time of task $\boldsymbol{n\ i}$
8. end do

9. Assign tasks $\boldsymbol{ni}$ to processor $\boldsymbol{Pj}$ in the most efficient way possible. Heterogeneous Finishing the Execution as soon as possible $\boldsymbol{ni}$.
10. end while

Priorities are defined as rank in the first phase task (u). $ranku(ni) = wi + maxnj \in succ(ni)\{ci,j + ranku(nj)\}$. reflects the longest task path length ni to the exit node, taking into account the computational cost of ni. rank $(nexit) = wexit$ for the exit task. The tasks are listed in order of diminishing ranku value. The task at the processor is in charge of the top of the task list pj that allows task ni .HEFT (Heterogeneous Earliest Finish Time) to be met during the processor selection phase.

$$\text{Timeconst(U)} = \frac{\text{time}_{\max} - \text{finiTime(U)}}{\text{time}_{\max} - \text{time}_{\min}} \quad (1)$$

The term $time_{max}$ in the formula denotes the maximum amount of time that the work may be completed in. The $time_{min}$variable indicates how long it will take to complete the task in the shortest possible time. finiTime (U) displays the task's actual completion time.

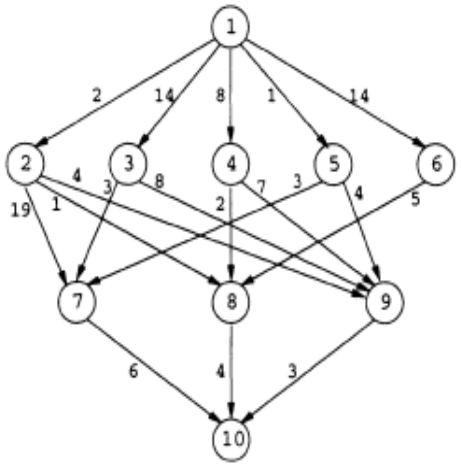

Figure 1: A Task Graph with Ten Tasks as an Example

Table 1. Graph HEFT Mapping Result

step	n_i	HEFT of n_i on p_i			Mapping
		P1	P2	P3	

1	n_1	10.00	11.0 0	11.00	P1
2	n_2	19.0	22.00	20.00	P1
3	n_5	32.00	33.00	24.00	P2
4	n_4	29.00	30.00	27.00	P3
5	n_3	27.00	30.00	35.00	P1
6	n_9	53.00	52.00	51.00	P3
7	n_7	38.00	47.00	61.00	P1
8	n_6	40.00	26.00	32.00	P2
9	n_8	39.00	31.00	33.00	P2
10	n_{10}	70.00	65.00	67.00	P3

The time $O(e\ x\ p)$ is used to compute the average values in step 1 of Fig 1; in step 2, the length of the critical path is estimated by $O(e)$, where e is the chart's edge. Step 3 takes $O(n\ log\ n)$ to specify tasks by their important path for n_i tasks. Steps 5-9 do not take a task if a job n_i has a task specified in its immediate predecessor or $O\ (a\ x\ p)$, hence the while loop takes a time $O.(e\ x\ p)$. As a result, the HEFT algorithm has a complexity of $O.(\ e\ x\ p)$.

3.2 TASK SCHEDULING ALGORITHM BASED ON HEFT

A task-planning algorithm for a limited number of processors in a heterogeneous technology environment. We use the HEFT to examine the complexity of time for the first time. The HEFT is a compound. DAGs are used to partition an application into tasks of varying sizes. A directed edge represents the dependency between tasks, and each task of a DAG corresponds to a sequence of actions. A DAG is represented by the graph $G\ =\ (V, E)$, where V represents the set of v tasks and E represents the set of e edges that connect the tasks.

Each edge $(i\ j)$ E denotes a dependency, such that task ni must finish before task nj can begin. If a task does not have a parent task, it is defined as the first task in a workflow. A task is defined as the exit task in a workflow of tasks if it has no children. In the case of DAG, a Matrix W is derived, i.e., the $v\ x\ p$ computer cost matrices, where v is the total number of works and where p represents the number of processors,$w_{i,j}$ indicates the anticipated execution time to perform job vi on processor vj. Equation defines the average execution time of task vi (2)

$$\overline{w_J} = \frac{\sum_{j\in P} w_{i,j}}{P} \tag{2}$$

Each edge $(i,j)\ \in E$ is associated with a non-negative weight ci,j which represents the communication cost between task $vi\ and\ vj$. The average communication cost of an edge (i,j) is defined in Equation (3)

$$\overline{c_{\iota,J}} = \bar{L} + \frac{Data_{i,j}}{\bar{B}} \tag{3}$$

$$makesapan = \max\{AFT(n_{exit})\} \tag{4 }$$

The make-span, also known as the schedule length, specifies the end time of the final task in the specified DAG, is a prominent statistic of the task planning. Equation (4) defines the Make-Time, which means that $AFT(n_{exit})$is the actual Finish Time of the exit node. Task t_i on resource $r_j$is the earliest time and end time as indicated under $EST(t_i, r_j)$ and $HEFT(T_i, r_j)$.

$$EST(t_i, r_j) = \max\{avail\ (t_i, r_j), \max_{t_m \in Pred(t_i)}\{AFT(t_m) + c_{mi}\},$$

$$HEFT(t_i, r_j) = w_{i,j} + EST(t_i, r_j), \tag{5}$$

Where avail (t_i, r_j) is the earliest time the resource r_j is ready for task execution, $AFT(t_m)$ shall be t_i and c_{mi} shall be edge communication costs from task t_m tasks to Task t_i. When the resource r_j is ready for work performance, the $AFT(t_m)$. All t_i must r_j immediate antecedent tasks were programmed before the calculation of the earliest completion time of a task $EST(t_{entry},\ r_j)\ =\ 0$ for the entry task t_{entry}.

4. EXPERIMENTAL RESULTS

A simulation method evaluates the proposed methodology. In order to evaluate our scheduling strategy, we employ the simulator that comprises working flow apps and modelling of the cloud environment using a CloudSim tool. A comparison study of certain state-of-the-art algorithms is conducted to evaluate the performance of the upgrading algorithm.

Number of tasks	HEFT	DEWTS	EES
0	35.0	36.09	39.67
3	42.67	45.78	50.34
6	53.67	57.89	62.67
9	67.89	71.89	75.67
12	77.78	81.67	88.67

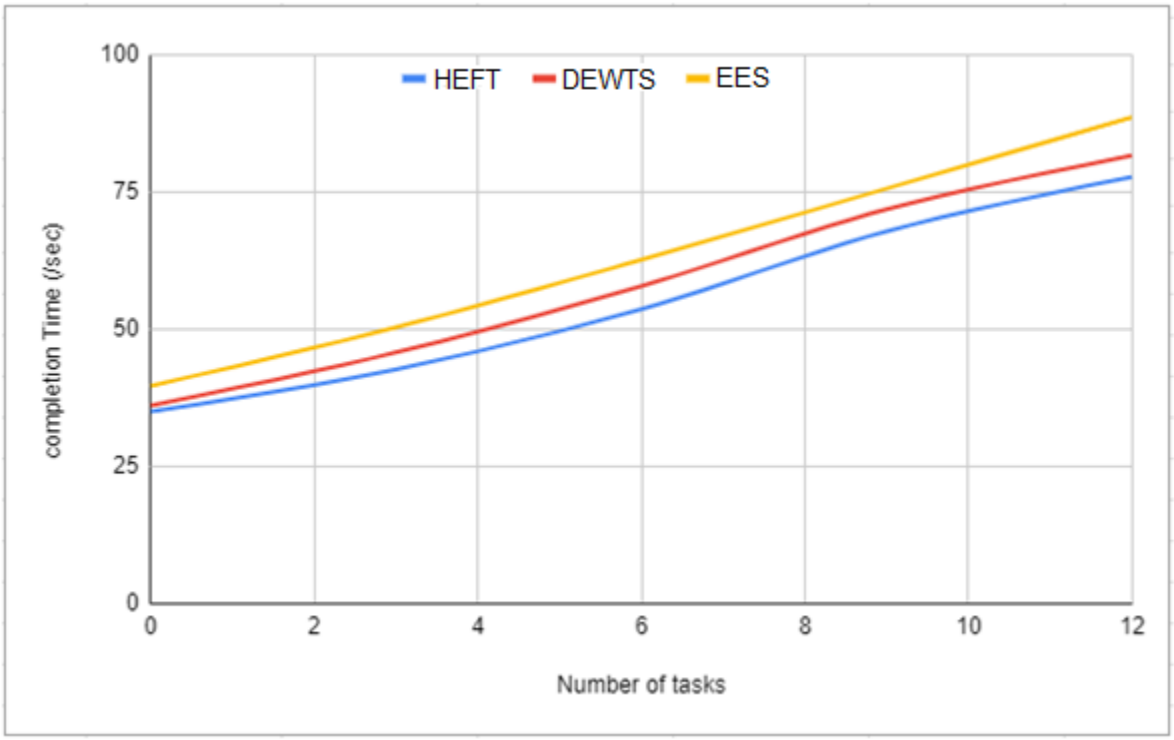

Figure 2: Completion time Graph Comparison

In Fig. 2, the HEFT algorithms are shown. In terms of the EES and DEWTS, the provided algorithm reduces completion time, as evidenced by the results. Furthermore, the results reveal that our technique is utilized to lower the time it takes for huge applications to complete. The proposed algorithm improves completion time based on analysis.

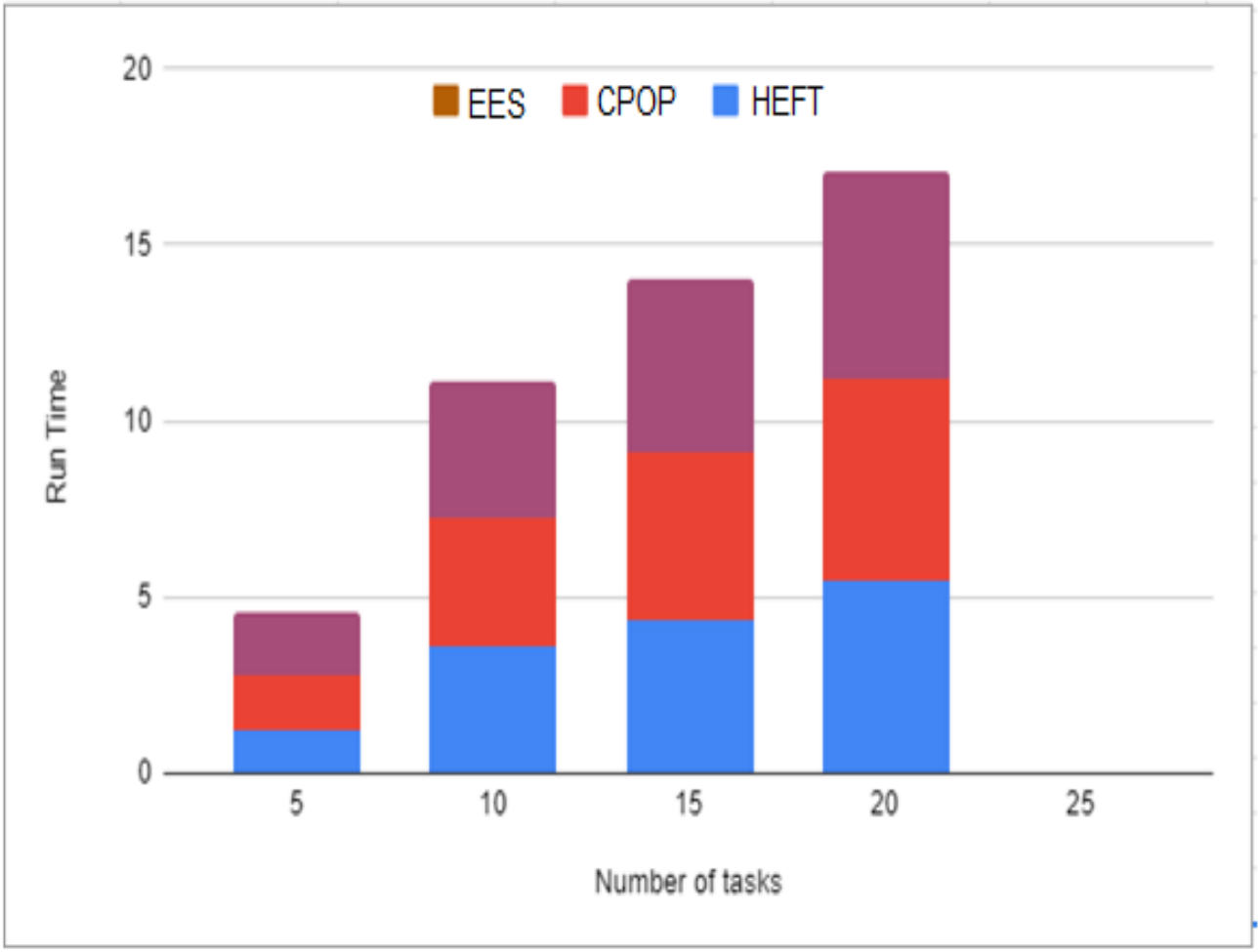

Figure 3: Runtime Graph Comparison

Figure 3 indicates that increasing the connectedness between nodes in the graph causes the algorithms to execute slower. The performance difference between HEFT and EES is due to how critical path works are handled in the total runtime. Because it uses a two-phase rank computation, the CPOP algorithm takes longer to run.

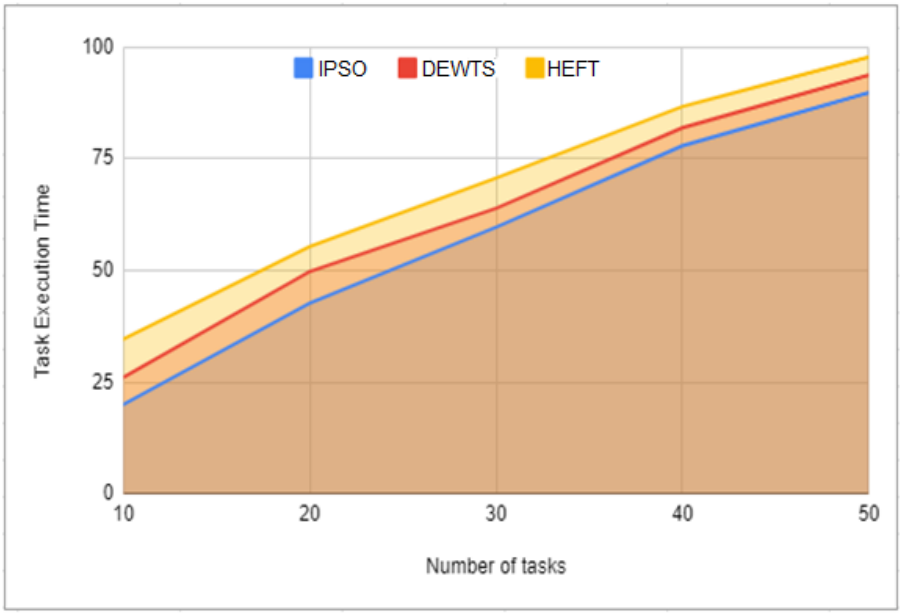

Figure 4: Task Execution Time Comparison Diagram

As illustrated in Figure 4, the length of time you need to use DEWTS algorithm and IPSO method in cloud computer planning is better in cloud environments. In addition, the efficiency advantage of HEFT is more visible due to the growth in the number of resources. The IPSO algorithm is not dynamically adaptable and efficient. The process of the selection of resources is omitted from the DEWTS algorithm, and local optimization is easy. The HEFT method fully addresses cloud computing dynamics and enhances task efficiency through resource selection and planning mechanisms in the implementation of the cloud computing.

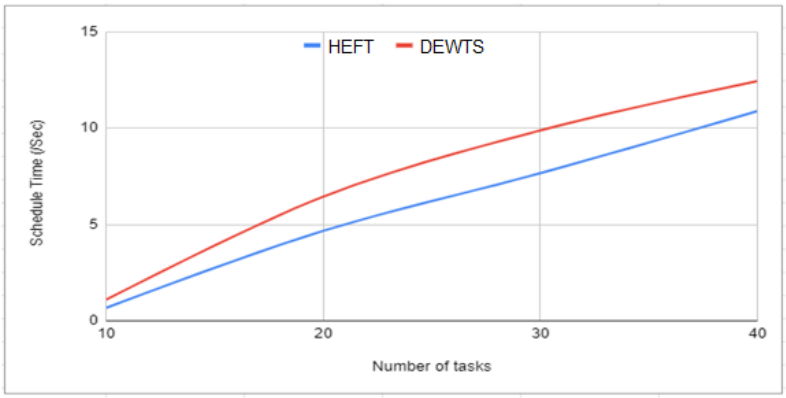

Figure 5: The proposed HEFT approach was compared to the existing DEWTS method.

Figure 5 depicts the schedule time results achieved using various strategies. The behavior of schedule time for HEFT is comparable to that obtained by DEWTS, as shown in the figure. The performance of both algorithms is similar for a small number of tasks and a low value of CCR. The schedule time is increased in DEWTS and HEFT algorithm achieve low schedule time.

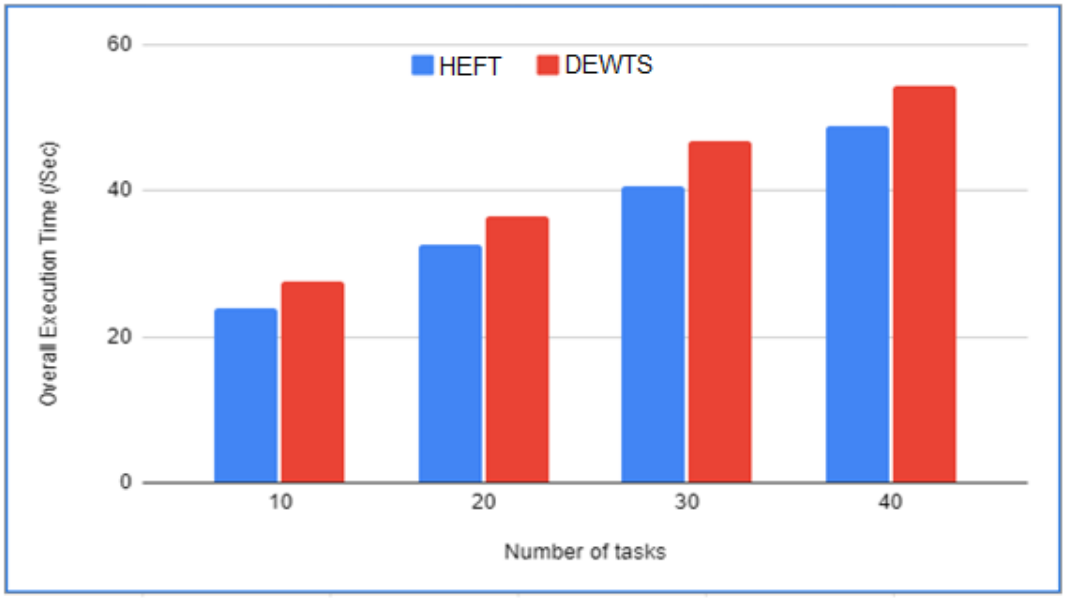

Figure 6: Comparison of overall execution time of HEFT and DEWTS

Figure 6 shows that performance in heterogeneous grid environments is quite essential. For both workflows, the performance guidance outcomes are almost two times better in the best scenario than those obtained without performance guidance. Evaluations of performance are certainly significant, although not extremely accurate.

5. CONCLUSION

This paper presents a task scheduling algorithm HEFT for heterogeneous computing systems. This new algorithm is a two-phase approach combining list-based, priority task and task planning processes. The HEFT method thus makes the scheduling of overall tasks more efficient. In the priority phase of the task, the choice of tasks is proposed for three levels of priority. The way to design tape queue not only takes crucial works into account but also takes the importance of parenting tasks into account. The replacement of parent works aims to cut communication expenses in the resource selection phase. The novel algorithm's performance is compared to that

of three of the finest existing scheduling algorithms: EES, DEWTS, and IPSO. The comparison study is based on a collection of task graphs and a randomly generated task graph. The HEFT algorithm provides the best performance in terms of speedup, execution time, runtime, and time optimization. In future, this work can be improved by focusing on load balancing issue.

REFERENCES

[1]. Chen, W., Xie, G., Li, R., & Li, K. (2020). Execution cost minimization scheduling algorithms for deadline-constrained parallel applications on heterogeneous clouds. Cluster Computing. doi:10.1007/s10586-020-03151-w

[2]. Maleki, N., Rahmani, A. M., & Conti, M. (2020). SPO: A Secure and Performance-aware Optimization for MapReduce scheduling. Journal of Network and Computer Applications, 102944. doi: 10.1016/j.jnca.2020.102944.

[3]. Dash, S. S., Das, S., & Panigrahi, B. K. (Eds.). (2021). Intelligent Computing and Applications. Advances in Intelligent Systems and Computing. doi:10.1007/978-981-15-5566-4

[4]. Shukla, R. K., Agrawal, J., Sharma, S., Chaudhari, N. S., & Shukla, K. K. (Eds.). (2020). Social Networking and Computational Intelligence. Lecture Notes in Networks and Systems. doi:10.1007/978-981-15-2071-6

[5]. Zhou, J., Zhang, M., Sun, J., Wang, T., Zhou, X., & Hu, S. (2020). DRHEFT: Deadline-Constrained Reliability-Aware HEFT Algorithm for Real-Time Heterogeneous MPSoC Systems. IEEE Transactions on Reliability, 1–12. doi:10.1109/tr.2020.2981419.

[6]. Hu, Y.-C., Tiwari, S., Trivedi, M. C., & Mishra, K. K. (Eds.). (2020). Ambient Communications and Computer Systems. Advances in Intelligent Systems and Computing. doi:10.1007/978-981-15-1518-7

[8]. Hosseinioun, P., Kheirabadi, M., Tabbakh, S. R. K., & Ghaemi, R. (2020). A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. Journal of Parallel and Distributed Computing. doi:10.1016/j.jpdc.2020.04.008 .

[9]. Liang, B., Dong, X., Wang, Y., & Zhang, X. (2020). A low-power task scheduling algorithm for heterogeneous cloud computing. The Journal of Supercomputing. doi:10.1007/s11227-020-03163-8 .

[10]. Han, P., Du, C., Chen, J., Ling, F., & Du, X. (2020). Cost and Makespan Scheduling of Workflows in Clouds Using List Multiobjective Optimization Technique. Journal of Systems Architecture, 101837. doi:10.1016/j.sysarc.2020.101837 .

[11]. Iranmanesh, A., & Naji, H. R. (2020). DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing. Cluster Computing. doi:10.1007/s10586-020-03145-8 .

[12]. Arulkumar, V., & Bhalaji, N. (2020). Performance analysis of nature inspired load balancing algorithm in cloud environment. Journal of Ambient Intelligence and Humanized Computing. doi:10.1007/s12652-019-01655-x .

[13]. Al-Turjman, F. (Ed.). (2020). Trends in Cloud-based IoT. EAI/Springer Innovations in Communication and Computing. doi:10.1007/978-3-030-40037-8 .

[14]. Aziza, H., & Krichen, S. (2020). A hybrid genetic algorithm for scientific workflow scheduling in cloud environment. Neural Computing and Applications. doi:10.1007/s00521-020-04878-8 .

[15]. Velliangiri, S., Karthikeyan, P., Arul Xavier, V. M., & Baswaraj, D. (2020). Hybrid electro search with genetic algorithm for task scheduling in cloud computing. Ain Shams Engineering Journal. doi: 10.1016/j.asej.2020.07.003 .

[16]. Zhang, L., Zhou, L., & Salah, A. (2020). Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments. Information Sciences, 531, 31–46. doi:10.1016/j.ins.2020.04.039 .

[17]. Meng, S., Huang, W., Yin, X., Khosravi, M. R., Li, Q., Wan, S., & Qi, L. (2020). Security-aware Dynamic Scheduling for Real-time Optimization in Cloud-based Industrial Applications. IEEE Transactions on Industrial Informatics, 1–1. doi:10.1109/tii.2020.2995348 .

[19]. Hou, S., Ni, W., Zhao, S., Cheng, B., Chen, S., & Chen, J. (2020). Decentralized Real-time Optimization of Voltage Reconfigurable Cloud Computing Data Center. IEEE Transactions on Green Communications and Networking, 1–1. doi:10.1109/tgcn.2020.2987063 .

[20]. Shukri, S. E., Al-Sayyed, R., Hudaib, A., & Mirjalili, S. (2020). Enhanced multi-verse optimizer for task scheduling in cloud computing environments. Expert Systems with Applications, 114230. doi:10.1016/j.eswa.2020.114230 .

[21]. Shirani, M. R., & Safi-Esfahani, F. (2020). Dynamic scheduling of tasks in cloud computing applying dragonfly algorithm, biogeography-based optimization algorithm and Mexican hat wavelet. The Journal of Supercomputing. doi:10.1007/s11227-020-03317-8 .

[22]. Fang, J., & Ma, A. (2020). IoT Application Modules Placement and Dynamic Task Processing in Edge-Cloud Computing. IEEE Internet of Things Journal, 1–1. doi:10.1109/jiot.2020.3007751 .

Authors' Profile

Lokesh Sivanandam received his B.E. Degree in Electrical and Electronics Engineering from Dr. MGR Engineering College and M.E. Degree in Electronics Engineering from Madras Institute of Technology, Chennai, India, and in the year 2002 and 2005 respectively. He obtained his Ph.D. Degree in the area "Low Power Testing of SoCs" in the year 2020 from Anna University, Chennai, India. He is currently Assistant Professor (Sr. Gr) in Ramanujan Computing Centre, College of Engineering, Guindy, Anna University, Chennai, India. His research interest is in the area of Testing of Digital Circuits and SOCs. He is a member of the Computer Society of India.

Sakthivel Periyasamy received a B.E. degree in Computer Science and Engineering from the University of Madras, India in 1992, an M.E degree in Computer Science and Engineering from Jadavpur University, India in 1994, and a Ph.D. degree in Computer Science and Engineering from Anna University, India in 2008. He is currently a Professor in the Department of Electronics and Communication Engineering, Anna University, India. He is a member of Computer Society of India (CSI), Institute of Electronics and Telecommunication Engineers (IETE), Institution of Engineers (India), Indian Society for Technical Education (ISTE), VLSI Society of India (VSI), Institute of Electrical and Electronics Engineers (IEEE), Association of Computing Machinery (ACM) and Institute of Electronics, Information and Communication Engineers (IEICE). His research interests include VLSI Design and Testing and Computer-Aided Design of VLSI Circuits.

Uma Maheswari Oorkavalan received her B.E. degree in Electronics and Communication Engineering and M.E. Degree in Communication Engineering from Thiagarajar College of Engineering, Madurai, India, in the year 1998 and 1999 respectively. She obtained her Ph.D. Degree in the area "Nonlinear signal processing" in the year 2009 from Anna University, Chennai, India. She is currently a Professor in the Department of Electronics and Communication Engineering, College of Engineering, Guindy, Anna University, Chennai, India. Her research includes nonlinear signal processing, underwater image enhancement, and analysis of FMRI. She is a senior member of IEEE and a life member of the Computer Society of India.