ENERGY MINIMIZATION AND TASK DEADLINE AWARE WORKLOAD SCHEDULING (EMTDA-WS)

Hrushikesh Joshi(1), Uttam Patil(2), Anand Diggikar(2)*

[1] Research Scholar at Jain College of Engineering, CSE Department, VTU, Belagavi, Faculty of Engineering, IT Department, SCTR's Pune Institute of Computer Technology, Pune
[2] Department of CSE, Faculty of Engineering, Jain college of Engineering, VTU, Belagavi
India

* Corresponding Author, e-mail: hjjoshi@pict.edu

Abstract: The execution of scientific workflow on cloud platform is time consuming and expensive. As users are charged based on hour of usage, lot of research work have been emphasized in minimizing processing time for reduction of cost. However, the processing cost can be reduced by minimizing energy consumption especially when resources are heterogeneous (i.e., multi-core edge platform) in nature; Very limited work have been done considering optimizing energy and processing time parameter together in meeting task Quality of Service (QoS) requirement. This paper presents an energy minimization and task deadline aware workload scheduling (EMTDA-WS) technique under heterogeneous cloud platform. Experiments are conducted using widely used workflow such as Montage. The outcome shows the EMTDA-WS significantly reduces energy, and processing time in comparison with standard workload scheduling model.

Key words: Cloud, Heterogeneous Computing, Quality of Service, Workload Scheduling, Montage Workflow.

1. INTRODUCTION

Mostly the cloud computing platforms are used to execute a workflow like a web-service as it requires a high-performance computation for the execution of the tasks. Moreover, the large workflow can be evaluated by the process of simulation in an effective way consuming less time and also by reducing the cost. Most of the scientific workflows are denoted using a Directed Acyclic Graph (DAG). In the DAG graph, the vertices are used to denote the dependencies and the edges are used to denote the set of tasks. Hence, the upcoming tasks will not be allocated the resources until and unless the existing tasks are completed. Therefore, the dependency of the tasks makes the scheduling in the cloud platform a challenging task. Moreover, to design an efficient workload scheduling we have to look at the challenges faced by the existing models like the time and cost required for the execution of the given task and their huge scientific workloads. Hence, due to this the cloud platform is not able to meet the deadline of the execution of the task. Many researchers are mainly focusing to find an optimal solution by utilizing some heuristic algorithms. Though, the heuristic algorithms used, consider only the execution of the task and do not consider the deadline required for the execution of the task. Due to this, the heuristic algorithms fail to attain an optimal solution and this affects the total QoS and increases the Service Level Agreement violation. Hence, the scheduling of any workflow is said to be a NP-Hard problem. To reduce the energy overhead and the deadline for the execution of the task it is a very challenging task. Most of the existing models have failed to utilize the Virtual Machine (VM) Selection Policy in the workflow scheduling during the designing process, hence due to this there exists the energy-makespan problem. The significance of the research has been described below as follows

- To design an energy efficient workload scheduling metric under multi-core edge-cloud platforms.
- To design a trade-off model to reduce the energy consumption and to meet all the task deadline prerequisite

2. LITERATURE SURVEY

In this section, we have surveyed various work in order to design a trade-off model which can reduce the energy consumption and also to meet the task deadlines. In [1], they have presented how the total consumption of energy affects in increasing of the cost-of-service provisioning. They have designed a scheduling framework which reduces the consumption of energy and also meets the timeliness and reliability for the proper execution of the workflow. In this model they have used the method of the Non-Linear-Mixed-Integer Programming to obtain an optimal solution. In [2], they have presented a model using trade-off to take care the problem of the unpredictable resource allocation in the cloud computing using an algorithm. In this model, they have used the multi-objective-parameter optimization technique in order to reduce the makespan and cost both at the same time. In [3], they have presented a model, Faster Nested Particle Swarm Optimization (FNPSO), which reduces the execution time of the workflow. The results show that the model shows better performance when compared with the existing NPSO model. In [4], they have utilized both the methods, Heterogeneous-Earliest-Finish-Time (HEFT) and Q-Learning (QL) to develop a workflow scheduling method called QL-HEFT. The model main intention was to reduce the computation and execution time. In this method the Q-Learning initially takes the jobs of the tasks and then searches a suitable machine in order to execute the task in a given deadline. In [5], they have presented a model to schedule the workflow by considering the contention awareness. In order to reduce the makespan, a heuristic model having an endpoint contention awareness has been developed. They have also redesigned the rescheduling design to improve and increase the efficiency of the workflow scheduling. In [6], they have presented a model, DCOH, to schedule the workflow by using the evolutionary computing method to reduce the computation time and meet the deadlines of each task. Furthermore, they have improved the DCOH by utilizing the multi-objective-parameter by reducing the cost and makespan in a cloud network. In [7], they have developed a model for scheduling workflow to meet the deadline of the task and reduce the cost. In this model, they have used the priority selection method to establish the job of the task and to allocate all the required necessary computational resources to that task in a given deadline. To improve the reliability, they have discarded some of the decisions using the discarding method. In [8], they have presented that for an efficient workflow it is necessary that both the SLA and the deadline requirement should be met. In this model they have used the multi-cloud network in order to reduce the cost and meet all the necessary performance requirement in the stream workflow applications. In [9], they have presented a fault-tolerant-scheduling method for the execution of the workflow in the multi-cloud network. Furthermore, this model guarantees that it meets all the required reliability aspects and also reduces the cost of the execution of the workflow. In [10], they have proposed an algorithm, energy efficient workflow scheduling (EEWS) using the existing Hybrid-Chemical-Reaction-Optimization (HCRO) method. This method has been proposed to reduce the energy consumption and also to minimize the makespan. In this model, the energy consumption is evaluated using the DVS-enabled environment. In [11], they have presented a hybrid algorithm by using the Artificial Bee Colony method and Genetic Algorithm for scheduling the multi-objective workflow in IaaS cloud. This method has been proposed to reduce the cost and makespan for the scheduling of the workflow in the IaaS cloud platform. In [12], they have proposed an algorithm on the basis of task threshold and task classification to allocate efficient resources for an efficient scheduling of the workflow. However, energy optimization under multi-core processing environment is not considered. As a result, failed to bring good balance between energy-makespan minimization meeting applications QoS prerequisite. In addressing the research problem in next section energy minimization and task deadline aware workload scheduling is designed considering multi-core edge-cloud computing environment.

3. ENERGY MINIMIZATION AND TASK DEADLINE AWARE WORKLOAD SCHEDULING (EMTDA-WS) MODEL

In this section, the CPU is considered as a normal multicore processor and the GPU has been considered as advanced multicore processor. This section gives a detail about the working of the normal multicore processor and advanced multicore processor, load optimization and reduction of the energy consumption. The proposed model, EMTDA model works on the idea of adaptive computing through the advanced multicore processor which utilizes a large number of virtual machines at a time having more central resource controller which is used to control them and a faster computation rate. When a job of a given task is allocated, the central resource allocator allocates the resources to the normal multicore processor and advanced multicore processor simultaneously and started the execution. To communicate inside the model, a message passing method is used. Our proposed model, EMTDA model mainly reduces the energy consumption and increases the performance during the task allocation. Our model utilizes the normal multicore processor and advanced multicore processor for the heterogeneous cloud computing.

3.1. Load Optimization

This section provides the model to reduce the task load during the allocation of the resources for the tasks. Hence, we split our EMTDA model in different execution stages. Each stage has a specified task load for the execution of the task. Moreover, each task loading operation is split into various sub-tasks. The execution stage can be of the similar set-type or can be of different set-type. For example, consider a job in which a given number of tasks have to be executed which is denoted using J. In this example, the tasks which have to be executed are disseminated into two different task set. Consider one task set as A and another task-set as B. The task-set A denotes the collection of sub-tasks β which has to be allocated to the advance multicore processor and the task-set B denotes the collection of the sub-tasks which has to be allocated to the normal multicore processor. The allocation of the computation of the tasks can be represented using the following equation,

$$J = J_A + J_B \tag{1}$$

$$J_A = \beta . J \tag{2}$$

$$J_B = (1 - \beta) . J \tag{3}$$

The collection of the sub-tasks which has been allocated to the advanced multicore processor can be represented using the β and ranges from the range $0 \leq \beta \leq 1$. The J_A and J_B represents the overall sub-tasks which have been allocated to the normal multicore processor and the advanced multicore processor. The computational rate can be represented as K for a given task-load J which helps us to abridge the overall computational facilities which has been by the normal multicore processor and advanced multicore processor. The K, computational rate is defined as the amount of time taken to complete the sub-tasks of the task-load J by the normal multicore processor and advanced multicore processor in one second. Consider an instance in which the various task-loads utilize different set-types to express their sub-tasks to represent their own computational units. Due to this instance, the K computational rate can have different values for different task-loads and can also remain constant for the sub-tasks having similar set-type of the task load only if there are enough processors to execute all the tasks.

Consider the computational rates as $K, K_A\ and\ K_B$ for a heterogenous system, a normal multicore processor as host and an advanced multicore processor. For this configuration, the time required to execute all the sub-tasks of the given tasks in the normal multicore processor and advanced multicore processor is represented using $M, M_B\ and\ M_A$. This can be represented using the following equations,

$$M_B = \frac{J_B}{K_B} \tag{4}$$

$$M_A = \frac{J_A}{K_A} \tag{5}$$

$$M = \uparrow \{M_B, M_A + M_D\} \tag{6}$$

The Equation 6 denotes that whenever the set of the sub-tasks has been uploaded to the advanced multicore processor, it holds an amount of delay which is represented using D. The amount of delay D is the amount of time essential to send the information to the processor and to activate the processor. Whenever $M_A \gg M_D$, there is no delay in time and the performance of the processor remains the same. To calculate the overall time required for execution, we consider all the sub-tasks which have been divided form the task-load, the overall time of the normal multicore processor, advanced multicore processor and the delay caused during the offloading. This can be described using the equation which is given as follows,

$$K = \frac{J}{M} = \frac{1}{\uparrow\left(\frac{(1-\beta)}{K_B}, \frac{\beta}{K_A} + \frac{M_D}{J}\right)} \tag{7}$$

If $M_A \gg M_D$, then the amount of delay M_D can be neglected and the above equation can be represented as follows

$$K = \frac{J}{M} = \frac{1}{\uparrow\left(\frac{(1-\beta)}{K_B}, \frac{\beta}{K_A}\right)} \tag{8}$$

3.2 Energy Consumption

This section provides an efficient model for the energy consumption of the normal multicore processors and the advanced multicore processors has been given. The energy of the whole model has been divided into three parts which is represented using the following equation,

$$E = \sum_{x \in (A,B,S)} E_x \tag{9}$$

In Equation 9, the normal multicore processor is represented using A and the advanced multicore processor is represented using B. The memory consumed by both the processors is represented using S. The overall energy consumption can be divided between the static energy $E_{\mathbb{S}}$ and dynamic energy $E_{\mathbb{D}}$. The static energy $E_{\mathbb{S}}$ is generated because there is no task for the execution of the task-load J and dynamic energy $E_{\mathbb{D}}$ is generated because of the execution of the task. The reason of the generation of the static energy $E_{\mathbb{S}}$ is because there is no task for execution so the speed of the processor remains constant until a task is offloaded for execution. The static energy can be combined to a single static energy consumption model using the following equation,

$$E_{\mathbb{S}} = \sum_{x \in (A,B,S)} E_{\mathbb{S}_x} \tag{10}$$

During the execution of the task, there are many reasons why there is more consumption of energy. The reasons are described below as follows,

1.For the overall execution time M, the model may utilize the static energy $E_{\mathbb{S}}$.

2.For the normal multicore processor M_B, the model may utilize the dynamic energy $E_{\mathbb{D}_B}$.

3.For the advanced multicore processor M_A, the model may utilize the dynamic energy $E_{\mathbb{D}_A}$.

4.The energy E_D essential to host advanced multicore processor by the host normal multicore processor with the energy $E_{\mathbb{S}_B}$, i.e., whenever an advanced multicore processor consumes more time when compared with the normal multicore processor like $M_A > M_B$.

From all the above reasons, the overall energy consumption for the above-mentioned reasons can be given represented using the following equation,

$$P = M.E_{\mathbb{S}} + M_B.E_{\mathbb{D}_B} + M_A.E_{\mathbb{D}_A} + (M_A - M_B).E_D \tag{11}$$

The above equation is considered as the overall system energy consumption because all the values of the components like the normal multicore processor and the advanced multicore processor can be evaluated directly. The overall energy efficacy γ can be described as the ratio of the task loads J to the overall energy P in task per joule. The ratio of the average energy consumption and average computation rate can be defined in G-Flops per watts or tasks per watt. The total efficacy γ can be represented using the given below equation utilizing the Equation 4 and Equation 11,

$$\gamma = \frac{J}{P} = E_{\mathbb{S}}.\uparrow\left(\frac{(1-\beta)}{K_B}, \frac{\beta}{K_A}\right) + \left\{\frac{(1-\beta).E_{\mathbb{D}_B}}{K_B}\right\} + \frac{\beta.E_{\mathbb{D}_A}}{K_A} + E_D. \\ \uparrow\left(\frac{\beta}{K_A} - \left\{\frac{(1-\beta)}{K_B}\right\}, 0\right) \tag{12}$$

The Equation 12 denotes the total energy efficacy of γ. The offloading time of delay M_D is rejected because of the M_A. Equation 11 is defined to explain the overall energy efficacy of γ, i.e., it can be defined using different constraints and factors like static energy of the system, processing distribution β, performance of the system and the energy distribution among the normal multicore processors and advanced multicore processors.

The efficiency of energy in the normal multicore processor and advanced multicore processor can be defined as γ_B and γ_A. The efficiency of normal multicore processor and advanced multicore processor can be defined as follows using the following equation

$$CPUs = \frac{K_B}{E_{\mathbb{D}_B}} \tag{13}$$

$$GPUs = \frac{K_A}{E_{\mathbb{D}_A}}. \tag{14}$$

Our proposed EMDTA model utilizes the normal multicore processor and advanced multicore processor. For the normal multicore processor, the system parameters $K_B, E_{\mathbb{S}_B}\ and\ E_{\mathbb{D}_B}$ are the utility of frequency of the normal multicore processor. Similar to the normal multicore processor, for the advanced multicore processor the system parameters $K_A, E_{\mathbb{S}_A}\ and\ E_{\mathbb{D}_A}$ are the utility of frequency of the advanced multicore processor. In this model, EMDTA, the processing rate and the energy consumption are the utility of β, $\mathbb{F}_B$ and $\mathbb{F}_A$. Hence, the utility functions like $K_A, K_B, E_{\mathbb{D}_B}$ and $E_{\mathbb{D}_A}$ differ when the frequency of the system varies.

3.2.1. Identifying Finest Performance Evaluation

Consider both the processors, normal multicore processor and advanced multicore processors execute the given task at their required speed. Our proposed EMTDA model provides more optimal processing distribution for the different constraints like performance, energy and energy efficiency. Both the energy efficiency and the performance of the model rely on the distribution factor β. In the advanced multicore processor, the best performance can be obtained when the normal multicore processor and the advanced multicore processor overlap each other i.e., $M_B = M_A$ or $\frac{(1-\beta).J}{K_B} = \frac{\beta.J}{K_A}$, due to this the evaluation of the best performance parameter can be given as follows,

$$\beta_{\mathbb{P}} = \frac{K_A}{K_B + K_A} \tag{15}$$

In Equation 15, the evaluation of the best performance parameter $\beta_{\mathbb{P}}$ helps to attain the best performance by $K = K_A + K_B$ at the least time period $\frac{J}{K_A + K_B}$. The total efficiency of energy can be obtained using the following equation,

$$\gamma = \frac{K_B + K_A}{E_{\mathbb{S}} + E_{\mathbb{D}_B} + E_{\mathbb{D}_A}} \tag{16}$$

3.2.2. Identifying Finest Energy Evaluation

The processors, normal multicore processor and advanced multicore processor have the ability to assign the tasks the required resources. To assign the resources to a given task load, the optimum minimal energy distribution β_g can be calculated using the following equation,

$$\min_{\beta_g} E_{\mathbb{S}}.\uparrow\left(\frac{1-\beta}{K_B},\frac{\beta}{K_A}\right)+\frac{(1-\beta).E_{\mathbb{D}_B}}{K_B}+\frac{\beta.E_{\mathbb{D}_A}}{K_A}+E_D.\uparrow\left(\frac{\beta}{K_A}-\frac{(1-\beta)}{K_B},0\right) \tag{17}$$

3.3.3. Identifying Finest Frequency Evaluation

The processors, normal multicore processor and advanced multicore processor work at different processing speeds according to the task. Consider the processing speed of the normal multicore processing as N_B and the processing speed of the advanced multicore processing as N_A. The overall speed of both the normal multicore processor and the advanced multicore processor can be considered as possible pairs $\mathbb{F}_A, \mathbb{F}_B$. For each of the pairs there is an optimal parameter for the evaluation of the best performance $\beta_{\mathbb{P}}$ and an optimal parameter for the evaluation of the energy β_g. To boost the energy efficiency and performance of the whole system, the optimal frequency distribution can be $(\mathbb{F}_A^*, \mathbb{F}_B^*, \beta^*)$, which can be attained using the following stages,

1.For each of the possible pair $(\mathbb{F}_A, \mathbb{F}_B)$, our proposed method EMTDA helps to search $K_\uparrow(\mathbb{F}_A, \mathbb{F}_B)$ and the corresponding $(\mathbb{F}_A, \mathbb{F}_B, \beta_{\mathbb{P}})$.

2.The maximal value from the $K_\uparrow(\mathbb{F}_A, \mathbb{F}_B)$ and state the respective $(\mathbb{F}_A^*, \mathbb{F}_B^*, \beta_{\mathbb{P}}^*)$ can be found using our EMDTA model.

3.For each of the possible pair $(\mathbb{F}_A, \mathbb{F}_B)$, our proposed method EMTDA helps to search $\gamma_\uparrow(\mathbb{F}_A, \mathbb{F}_B)$ and the corresponding $\left(\mathbb{F}_A, \mathbb{F}_B, \beta_g\right)$.

4.The maximal value from the $\gamma_\uparrow(\mathbb{F}_A, \mathbb{F}_B)$ and state the respective $\left(\mathbb{F}_A^*, \mathbb{F}_B^*, \beta_g^*\right)$ can be found using our EMDTA model.

4. RESULTS AND DISCUSSION

In this section the results have been discussed using Montage workflow in our Energy Minimization and Task Deadline Aware Workload Scheduling (EMTDA-WS) model. In our proposed model we have considered various jobs like Montage 25, 50, 100 and 1000 to evaluate the execution time of our EMTDA-WS with the existing Dynamic Voltage and Frequency Scaling (DVFS) [1], Fault-Tolerant Cost-Efficient Workflow Scheduling Algorithm (FCWS) [9], Particle Swarm Optimization (PSO) [12] model.

4.1. Montage Workflow

The workflow of the Montage has been given in the Figure 1. In this Montage workflow, each network node defines a level of action in the workflow. Each level has a specific task. The initial level of the montage workflow validates only single action. The other levels have different action which are validated in each of the network node. Every network node in the workflow uses the same executable code in which the input is divided and each node considers an input and the tasks are executed.

4.2. Execution Time

In this section, the execution time taken for executing the different Montage workflows keeping the virtual machines constant i.e., Virtual Machines=30 has been evaluated. The Figure 2 shows the execution-time of our model compared with the DVFS model [1]. The Figure 2 shows that our model has reduced the execution time when compared with the existing DVFS model. Moreover, for the Montage 25, Montage 50 and Montage 100 our EMTDA has showed better improvement by reducing the execution time. For the Montage 1000, our EMTDA has reduced by 90%. This shows our model is efficient in reducing the execution-time.

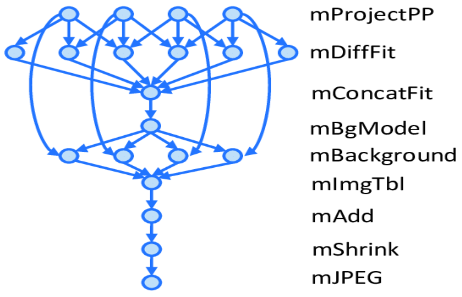

Figure 1: Montage Workflow

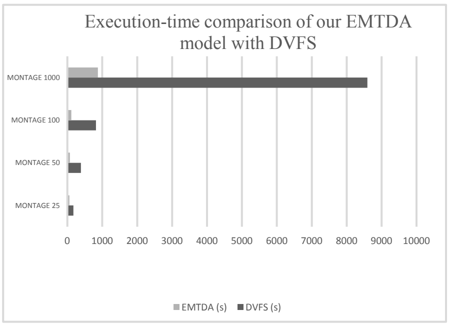

Figure 2: Execution-time comparison of our EMTDA model with DVFS

4.3. Power Sum

In this section, the Power Sum taken for executing the different Montage workflow keeping the virtual machines constant i.e., Virtual Machine=30 has been evaluated. The Figure 3 shows that our model has reduced the Power Sum when compared with the existing DVFS [1] model. Moreover, for the Montage 25, Montage 50 and Montage 100 our EMTDA has showed better improvement by reducing the power sum.

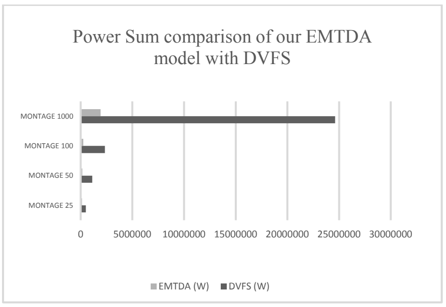

Figure 3: Power Sum comparison of our EMTDA model with DVFS

For the Montage 1000, our EMTDA has reduced by 92%. This shows our model is efficient in reducing the power-sum.

4.4. Power Consumption

In this section, the Power Consumption used for executing the different Montage workflow keeping the virtual machines constant i.e., Virtual Machine=30 has been evaluated. The Figure 4 shows that our model has reduced the Power Consumption when compared with the existing DVFS [1] model. Moreover, for the Montage 25, Montage 50 and Montage 100 our EMTDA model consumes very less power when compared with the existing model. For the Montage 1000, our EMTDA has reduced by 95%.

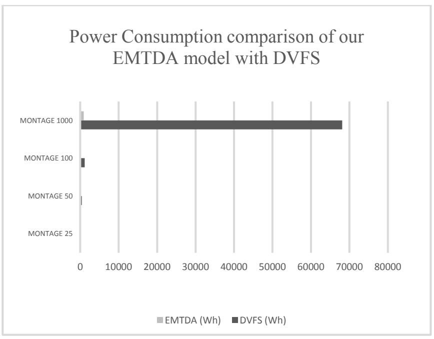

Figure 4: Power Consumption comparison of our EMTDA model with DVFS

This shows our model is efficient in reducing the power consumption.

4.5. Average Power Consumption

In this section, the Average Power Consumption utilized for executing the different Montage workflow keeping the virtual machines constant i.e., Virtual Machine=30 has been evaluated. The Figure 5 shows that our model has reduced the Average Power Consumption when compared with the existing DVFS [1] model. Moreover, for the Montage 25, Montage 50 and Montage 100 our EMTDA model consumes average less power when compared with the existing model. For the Montage 1000, our EMTDA has reduced by 10%. This shows our model is efficient in reducing the average power consumption.

4.6. Comparison of execution time with other models

In this section, we have compared the time required for the overall execution with the other existing models. In Figure 6, the Average Execution Time to run the different Montage workflow keeping the Virtual Machines as 30 for the DVFS [1] model and the PSO [12] model have been compared with our proposed EMTDA model. The results show that our EMTDA model takes less time to execute the workflow in all the Montage workflows.

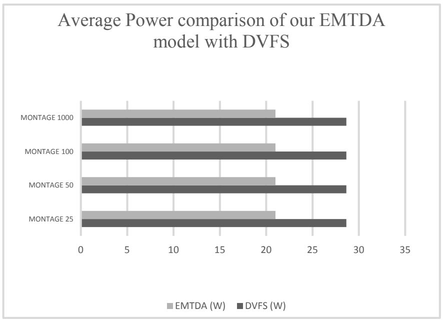

Figure 5: Average Power comparison of our EMTDA model with DVFS

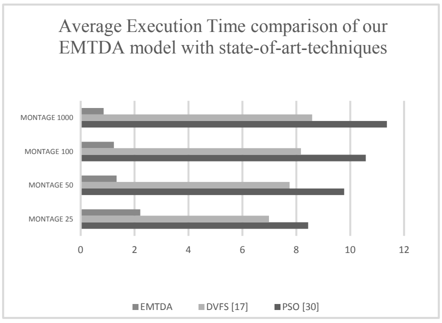

Figure 6: Average Execution Time comparison of our EMTDA model with state-of-art-techniques

4. CONCLUSION

In this paper, we have discussed our model Energy Minimization and Task Deadline Aware Workload Scheduling (EMTDA-WS). We have utilized the normal multicore processor and the advanced multicore processor to reduce the energy consumption and meeting the task deadline requirement. Our model distributes the task to the normal multicore processor and advanced multicore processor to minimize the offloading of the tasks and also to provide better resources for the tasks. In this paper, a model has been presented to minimize the offloading of the tasks and also to reduce the consumption of energy which is the biggest issue in the existing models. Moreover, another model to increase the energy efficiency and to provide better performance for our model has been presented. The advanced multicore processor provides better and faster results for our EMTDA-WS model and the normal multicore processor minimizes the high energy consumption. The experiments have been conducted on the Montage workflow and compared with the existing Dynamic Voltage and Frequency scaling (DVFS) model and other existing models. The attained results show that our model provides better results when compared with the existing models. Moreover, our model provides better performance in terms of execution time, power sum, energy consumption and average power consumption. For the future work, the model can be improved to reduce the overall cost for the execution of the workflow.

REFERENCES

[1] B. Hu, Z. Cao and M. Zhou. Energy-Minimized Scheduling of Real-Time Parallel Workflows on Heterogeneous Distributed Computing Systems. *IEEE Transactions on Services Computing*, 2021. doi: 10.1109/TSC.2021.3054754.

[2] T. Pham and T. Fahringer. Evolutionary Multi-objective Workflow Scheduling for Volatile Resources in the Cloud. *IEEE Transactions on Cloud Computing*, 2020 doi: 10.1109/TCC.2020.2993250.

[3] A. Song, W. -N. Chen, X. Luo, Z. -H. Zhan and J. Zhang, "Scheduling Workflows With Composite Tasks: A Nested Particle Swarm Optimization Approach, *IEEE Transactions on Services Computing*, vol. 15, no. 2, 2022, pp. 1074-1088, doi: 10.1109/TSC.2020.2975774.

[4] Tong, Zhao & Deng, Xiaomei & Chen, Hongjian & Mei, Jing & Liu, Hong. QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment. *Neural Computing and Applications*. Vol 32, 2020, pp. 1-18. doi:10.1007/s00521-019-04118-8.

[5] Q. Wu, M. Zhou and J. Wen, "Endpoint Communication Contention-Aware Cloud Workflow Scheduling, *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, 2022, pp. 1137-1150, doi: 10.1109/TASE.2020.3046673.

[6] Zhou, Junlong & Wang, Tian & Cong, Peijin & Lu, Pingping & Wei, Tongquan & Chen, Mingsong. Cost and Makespan-Aware Workflow Scheduling in Hybrid Clouds. *Journal of Systems Architecture*. 2019. pp. 100. doi:10.1016/j.sysarc.2019.08.004.

[7] G. Wang, Y. Wang, M. S. Obaidat, C. Lin and H. Guo, "Dynamic Multiworkflow Deadline and Budget Constrained Scheduling in Heterogeneous Distributed Systems, *IEEE Systems Journal*, vol. 15, no. 4, 2021, pp. 4939-4949, doi: 10.1109/JSYST.2021.3087527.

[8] M. Barika, S. Garg, A. Chan and R. N. Calheiros, "Scheduling Algorithms for Efficient Execution of Stream Workflow Applications in Multicloud Environments, *IEEE Transactions on Services Computing*, vol. 15, no. 2, 2022, pp. 860-875, doi: 10.1109/TSC.2019.2963382.

[9] X. Tang. Reliability-Aware Cost-Efficient Scientific Workflows Scheduling Strategy on Multi-Cloud Systems. *IEEE Transactions on Cloud Computing*, 2021. doi: 10.1109/TCC.2021.3057422.

[10] Singh V, Gupta I, Jana PK. An energy efficient algorithm for workflow scheduling in iaas cloud. *Journal of Grid Computing*. 2019, pp.1–20. https://doi.org/ 10.1007/s10723-019-09490-2

[11] Gao Y, Zhang S, Zhou J. A hybrid algorithm for multi-objective scientific workflow scheduling in iaas cloud. *IEEE Access*. Vol. 7, pp. 125783–125795.

[12] Malik, Nimra, Muhammad Sardaraz, Muhammad Tahir, Babar Shah, Gohar Ali, and Fernando Moreira. Energy-Efficient Load Balancing Algorithm for Workflow Scheduling in Cloud Data Centers Using Queuing and Thresholds. *Applied Sciences*. Vol. 11, no. 13, 2021, pp. 5849. https://doi.org/10.3390/app11135849.

Information about the authors:

Hrushikesh. J. Joshi received his BE degree in Computer Science and Engineering from Shivaji University and Master's degree (MTech) in Computer Engineering from BVCOE, Pune. He is a Ph.D. candidate with the department of Computer Science & Engineering at Jain college of Engineering, VTU, Belagavi, India.

Dr. Uttam Patil received the B.E. and MTech. degree form Visvesvaraya Technological University (VTU) in 2008 and 2012 respectively, Ph.D. in Faculty of Computer and information sciences form VTU in 2020. He is presently an Associate Professor and Head of the department of Computer Science and engineering in Jain College of Engineering Belgaum, Karnataka, India.

Dr. Anand N. Diggikar received the BE from Visvesvaraya Technological University (VTU) and Master of Computer Science from University of Auckland, New Zealand, Ph.D. in Faculty of Computer and information sciences form VTU. He held the post of Professor in KLE College of Engineering and Technology, Belgaum, Karnataka, India.

Manuscript received on 04 March 2022