

Optimal task partitioning to minimize failure in heterogeneous computational platform

Divyaprabha Kabbal Narayana, Sudarshan Tekal Subramanyam Babu

Department of Computer Science and Engineering, PES University, Bangalore, India

Article Info

Article history:

Received Mar 8, 2024

Revised Aug 17, 2024

Accepted Sep 3, 2024

Keywords:

Deadline

Energy

Heterogeneous computing

Parallel workflow application

Task failure

Workload scheduling

ABSTRACT

The increased energy consumption by heterogeneous cloud platforms surges the carbon emissions and reduces system reliability, thus, making workload scheduling an extremely challenging process. The dynamic voltage-frequency scaling (DVFS) technique provides an efficient mechanism in improving the energy efficiency of cloud platform; however, employing DVFS reduces reliability and increases the failure rate of resource scheduling. Most of the current workload scheduling methods have failed to optimize the energy and reliability together under a central processing unit-graphical processing unit (CPU-GPU) heterogeneous computing platform; As a result, reducing energy consumption and task failure are prime issues this work aims to address. This work introduces task failure minimization (TFM) through optimal task partitioning (OTP) for workload scheduling in the CPU-GPU cloud computational platform. The TFM-OTP introduces a task partitioning model for the CPU-GPU pair; then, it provides a DVFS-based energy consumption model. Finally, the energy-load optimization problem is defined, and the optimal resource allocation design is presented. The experiment is conducted on two standard workloads namely SIPHT and CyberShake workload. The result shows that the proposed TFA-OTP model reduces energy consumption by 30.35%, reduces makespan by 70.78% and reduces task failure energy overhead by 83.7% in comparison with energy minimized scheduling (EMS) approach.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Divyaprabha Kabbal Narayana

Department of Computer Science and Engineering, PES University

Bangalore, India

Email: divyaprabhamadhu@gmail.com

1. INTRODUCTION

Cloud computing platforms offer scalable computational resources of both software and hardware resources to its users on the go over the internet. In recent time, the cloud computing platform has started to use high-performance computing (HPC) like Apache Spark and Hadoop. to execute the data-intensive task in parallel [1]. The HPC allows the cloud provider to improve its resource utilization and also reduces the overall cost of workload execution of the client, thus, aiding in improving workload execution efficiency and users experience. The HPC platform must include a large cluster of central processing units (CPUs), and graphical processing units (GPUs) [2] and hardware to support different kinds of workload applications such as image processing and genome sequencing. for computation requirements. Each submitted task of the aforementioned application might request for different heterogeneous CPU-GPU resources; which makes workload scheduling a challenging task.

Efficient scheduling workloads are a crucial challenge in distributed computational platforms such as cloud computing, which often comprise diverse computational resources [3]. The objective of workload

scheduling is to allocate resources to the task while meeting specific constraints to minimize the makespan, as well as determining the initialization and completion times for each task. An effective design of workload scheduling techniques plays a critical role in achieving enhanced performance on heterogeneous HPC systems. The selection of a suitable scheduling technique can significantly reduce overall execution time, particularly when executing directed acyclic graph (DAG) applications that may have task dependencies. These systems use multiple types of computing resources with varying architecture, memory, processing, and speed, resulting in a nonpolynomial deterministic problem when scheduling workload for dependent tasks of respective DAG on the HPC environment in establishing the minimum cumulative makespan [4]. To address this issue in heterogeneous cloud computational environments, various methods have been developed, such as workload partitioning, optimal load balancing, and task migration across physical machines or multiple clouds, all of which aim to minimize the makespan of DAG applications running on virtual machines (VMs) [5].

The state-of-the-art HPC [6] are predominantly focused on achieving enhanced performance; thus, providing higher cluster size with higher processor frequency and multicore capability. Alongside, GPUs are used to improve parallel computational efficiency at the cost of higher energy consumption measured in performance/watt [7]. Thus, the current HPC systems are focused on designing energy-efficient HPC systems [8]. Additionally, the increasing electricity cost further motivated the researcher to develop a performance-oriented and energy-efficient design [7], [8]. Concerning these developments, this work aims to investigate the optimization of energy-aware scheduling and workload partitioning in the field of high-performance computing. To achieve this, we analyze several aspects, including energy-aware metrics [8], [9], system types about their heterogeneity, and compute device types, such as multi-core CPUs and many-core CPUs and accelerators such as GPUs. Additionally, we will examine the algorithmic methodologies used to address energy-aware [8], as well as energy-reliability scheduling problems [9], [10]. This work introduces task failure minimization (TFM) through an optimal task partitioning (OTP) scheme in a heterogeneous cloud computing platform. The model is energy efficient as well as performance-oriented leveraging CPU-GPU task partitioning and optimal scheduling design meeting energy-performance tradeoffs constraint.

The significance of the proposed work is as follows. First, a task partitioning model for execution tasks in both CPU and GPU is discussed. Then a dynamic voltage-frequency scaling (DVFS) scheme and energy consumption model for CPU-GPU system are employed. Introduced a tradeoff optimization problem between task traffic partitioning and speed scaling. Later, the optimal scheduling design of processor frequency-scaling optimization and energy-load optimization is modeled. The experiment result shows the TFM-OTP is very effective in executing the task within a given deadline meeting energy-load constraint.

The paper is organized as follows. In section 2, an extensive survey has been conducted to identify the significance of limitations of existing workload scheduling methods. In section 3 research methodology of task failure minimization using optimal task partitioning model is presented. In section 4 the results achieved in terms of makespan and energy consumption have been studied and comparative study has been provided with state-of-art workload scheduling techniques. The last section provides the significance in terms of percentage enhancement over state-of-the-art workload scheduling techniques.

2. LITERATURE SURVEY

In this section, different state-of-the-art methodologies related to task partitioning, workload prioritization, optimal scheduling and evolutionary models for scheduling workload in heterogeneous computational platforms are surveyed. Konjaang and Xu [11] introduced a multi-objective workload optimizer strategy (MOWOS) by focusing on reducing the cost [12] with a decent makespan and meeting complex scientific workload deadline prerequisites. The authors proposed a new virtual machine selection technique that works well for larger workloads; however, the model is tested only for homogenous computational models. Tang [13] introduced a reliability-aware cost-efficient scheduler (RACES) for executing scientific workflow in a multi-cloud platform. Then, studied the importance of reducing failure and providing a fault tolerant scheduling mechanism; In providing fault tolerance they proposed a multi-cloud-based resource scheduling design [14]. A probability density function for failure rate is used to provide reliability with minimal cost. However, because of poor load-balancing the model exhibits higher cost.

Kumar *et al.* [15] proposed autonomic resource provisioning and scheduler (ARPS) using a cloud platform by enabling an automatic decision-making model through the spider monkey optimization technique. The model focused on optimizing both cost and time meeting task deadline spider monkey optimization technique. The model reduced energy, cost, and time. The performance of such a model for scheduling complex scientific workflow execution is not explored. Calzarossa *et al.* [16] proposed a deadline and budget-awareness workload scheduler (DBAWS) by employing a multi-objective parameter optimization for a cloud environment with high uncertainty. They optimized both budgets and met task deadlines. The optimization problem is solved through Monte Carlo and the genetic algorithm. Qin *et al.* [17] introduced

knowledge adaptive discrete water wave optimizer (KADWWO) to investigate how cost can be kept low considering larger workflow application deadlines. They introduced a discrete water wave optimization strategy to adaptively reduce idle time with better load optimization attaining better convergence; thereby enhancing scheduling performance considering a highly dynamic cloud platform. Qin *et al.* [18] proposed a reliable-aware multiple parameter optimized memetic algorithm (RAMPOMA) by introducing a multi-objective optimization mechanism using parameters such as cost, reliability, and makespan for workflow execution in a multi-cloud heterogeneous platform. A memetic algorithm with a new genetic operator is introduced to provide better resource differentiation and backup strategies and these strategies are used to provide reliability; thereby enhancing the resource utilization of the network. Wang *et al.* [19] showed that workflow heterogeneity and resource heterogeneity will make scheduling a very challenging task. The authors proposed resource provision for multi-workflow for vertical and horizontal resource scaling using a reinforcement learning (RL) algorithm. Depth-first search-based coalition RL (DFSCRL) was used for better resource scaling optimization dynamically. Hu *et al.* [20] employed an energy-minimized scheduler (EMS) by employing dynamic voltage frequency scaling to provide energy-efficient resource scheduling [21]; a heuristic optimization is designed to assure reliability through mixed-integer programming. Though the model reduced energy consumption it did not consider energy-makespan optimization.

In Table 1, a comparative study of various existing and proposed models has been provided. The survey shows that extensive work has been reported on addressing the cost-makespan problem whereas very limited work has been reported on workload scheduling in a heterogeneous environment considering CPU-GPU optimization [22], [23]. Further, in comparison to cost-makespan very limited work has been done energy-makespan assuring reliability *i.e.*, minimal task failures leveraging CPU-GPU computing platform [24], [25]. In the next section, a new technique is proposed for energy-makespan with minimal task failure in meeting task deadline requirements.

Table 1. Comparative study

| | MOWOS (2021) [11] | ARPS (2021) [15] | DBAWS (2021) [16] | EMS (2022) [20] | KADWWO (2023) [17] | RAMPOMA (2023) [18] | DFSCRL (2023) [19] | TFM-OTP [proposed] |
|-------------------------|----------------------------|----------------------------|---------------------------------|--------------------|----------------------------------|---------------------------------|-----------------------|--|
| Heterogeneous computing | √ | × | × | √ | × | √ | √ | √ |
| Workload type | Complex | Simple | Complex | Complex | Complex | Complex | Complex | Complex |
| Workload size | Small to large | Small to medium | Small to medium-large | Small to large | Small to large | Small to large | Small to large | Small to large |
| QoS metrics | Time, budget and deadlines | Cost, time, and energy | Deadline and budget | Energy | Cost-makespan | Makespan, cost, and reliability | Makespan | Energy, deadline, resource utilization |
| Optimization strategy | Heuristic | Spider monkey optimization | Monte Carlo & Genetic Algorithm | Heuristic | Discrete water wave optimization | Memetic genetic algorithm | Q-learning | Optimal |
| Task Partitioning | no | no | no | no | no | no | no | yes |
| Reliability | no | no | no | yes | no | yes | no | yes |

3. PROPOSED METHOD

This section introduces task failure minimization through optimal task partitioning schemes for heterogeneous platforms. First provides details of workload and task Partition model. Second, provides details of the energy consumption model using DVFS. Finally, the tradeoff problem model is presented to attain optimal scheduling performance for the execution of parallel workflow applications.

3.1. Workload and task partition model

The data-intensive workload G is represented as a directed acyclic graph (DAG) composed of X sub-task with task-interdependencies E is expressed as (1).

$$G = \{X, E\} \quad (1)$$

The sub-task X can be defined by (2),

$$X = \{x_1, x_2, \dots, x_n\} \quad (2)$$

The sub-task x_j requests for operational frequency \mathbb{F}_j based on floating-point operations for executing sub-task x_j with a certain size \mathbb{K}_j . The task interdependencies are expressed as in (3),

$$E = \{e_{jl}(s_k, s_m)\} \quad (3)$$

The communication efficiency relies on the processing capability of CPU-enabled VMs and GPU-enabled virtual machines (VMs), $e_{jl}(s_k, s_m)$ defines the function associated with x_j 's processor s_k and x_l 's processor s_m . A sub-task without any prior task and a forthcoming task are defined as x_{in} and x_{out} , respectively. The workload G overall time can be measured using (4),

$$G = \mathbb{F}_j \times \mathbb{K}_j \quad (4)$$

In this work heterogeneous (*i.e.*, CPU-GPU) computational platform has been considered; thus, the clock frequency measured in cycles per second of CPU $f_j^{(c)}$ and GPU $f_j'^{(c)}$ is represented as (5),

$$f_j = f_j^{(c)} \times \mathbb{f}_j \quad (5)$$

where \mathbb{f}_j defines the number of CPU floating point operations per cycle. Similarly, the clock frequency measured in cycles per second of CPU $f_j^{(c)}$ and GPU $f_j'^{(c)}$ is represented by (6),

$$f_j' = f_j'^{(c)} \times \mathbb{f}_j' \quad (6)$$

where \mathbb{f}_j' defines the number of GPU floating point operations per cycle. Using a local gradient process to achieve data parallelism the input workload G is usually partitioned and executed in GPU or CPU. The partitioned workload executed in CPU considering node j is defined by the parameter G_j and for GPU is defined through parameter G_j' with constraint can be defined as in (7),

$$G_j + G_j' = G, \quad \forall j \in J. \quad (7)$$

Using the above partitioning scheme according to available frequencies of CPU and GPU, the local makespan for executing the workload G is given as (8).

$$u_j' = \max\left\{\frac{G_j}{f_j}, \frac{G_j'}{f_j'}\right\}, \quad \forall j \in J. \quad (8)$$

3.2. Dynamic voltage frequency and energy consumption model

Using the clock cycles of the processor the energy consumption can be measured as (9),

$$Q = \beta [f^{(c)}]^3 \quad (9)$$

where β is dependent on chip architecture and is measured in Watt/(cycle/s)³ and $f^{(c)}$ defines the processor clock frequency. The energy consumption of the CPU at node J is obtained in (10)

$$Q_j^{CPU} = \beta_j^{CPU} (f_j^{(c)})^3 = \mathcal{C}_j f_j^3 \quad (10)$$

where the parameter \mathcal{C}_j is obtained in (11),

$$\mathcal{C}_j = \frac{\beta_j^{CPU}}{\mathbb{f}_j^3} \quad (11)$$

Similarly, the energy consumption of GPU at node J is obtained by (12).

$$Q_j^{GPU} = \beta_j^{GPU} (f_j'^{(c)})^3 = \mathcal{G}_j f_j'^3, \quad (12)$$

where the parameter \mathcal{G}_j is obtained in (13),

$$\mathcal{G}_j = \frac{\beta_j^{GPU}}{\mathbb{f}_j'^3}. \quad (13)$$

The equations (9) to (13) are used for controlling the voltage-frequency scaling in a dynamic manner by optimizing the power of CPU and GPU through processing frequency speed f_j and f'_j control, respectively. The parameter C_j and G_j in (11) to (13) defines the computational efficiency with minimal task failures of heterogeneous platforms defined through the power-speed trade-off optimization function. In general, GPU-based communication plays significant computational benefits and CPU-based provides a supporting computational resource. The makespan G_j/f_j for CPU for completing the tasks of workload G_j within the given deadline and its energy consumption of respective node j is measured as in (14),

$$\mathcal{E}_j^{CPU} = C_j G_j f_j^2 \quad (14)$$

Similarly, the makespan G'_j/f'_j for GPU for completing the tasks of workload G_j within a given deadline and its energy consumption of respective node j is measured using (15),

$$\mathcal{E}_j^{GPU} = G_j G'_j f_j'^2. \quad (15)$$

Therefore, combining both CPU and GPU energy consumption the total energy consumption of corresponding node j is measured as (16),

$$\mathcal{E}_j^{cmp} = C_j G_j f_j^2 + G_j G'_j f_j'^2, \quad \forall j \in J. \quad (16)$$

3.3. Tradeoff problem definition

Designing effective task traffic partitioning and processing frequency speed scaling optimization which can aid in better management of distributed computational resources with minimal task failures and minimal overall energy consumption, the tradeoff functions are defined as (17),

$$\min_{\{G_j, f_j, f'_j\}} \sum_{j=1}^J (C_j G_j f_j^2 + G_j G'_j f_j'^2) \quad (17)$$

such that,

$$G_j + G'_j = \mathcal{E}, \quad G_j \geq 0, \quad G'_j \geq 0, \quad \forall j \in J, \quad (18)$$

$$u'_j = \max \left\{ \frac{G_j}{f_j}, \frac{G'_j}{f'_j} \right\}, \quad \forall j \in J, \quad (19)$$

$$0 \leq u'_j \leq U'_j, \quad \forall j \in J. \quad (20)$$

The constraint defined from (18), (19), and (20) satisfies (7), (8), and the time constraint defined through parameter U'_j defines the permissible maximum makespan of corresponding computational node j .

3.4. Optimal scheduling design

In providing better resource optimization design this section presents two optimization processes namely processor frequency-scaling optimization and energy-load optimization design. The work first introduces condition 1, for processor frequency-scaling optimization. Condition 1, focuses on reducing overall energy consumption, the processing speed of CPU-based f_j and GPU-based f'_j , the platform must be scaled as defined in (21),

$$\frac{G_j}{f_j} = \frac{G'_j}{f'_j}, \quad \forall j \in J, \quad (21)$$

considering respective random workload pair (G_j, G'_j) with $G = G_j + G'_j$. In obtaining optimal performance considering processor frequency-scaling according to the workload defined in condition 1 is instinctive and the outcome of equalizing GPU and CPU makespan to avoid non-performing one may end up with a bottleneck during computation process. Thus, the (17) is modified to follow the optimization problem.

$$\min_{\{G_j, u'_j\}} \sum_{j=1}^J \frac{C_j G_j^3 + G_j G_j'^3}{u_j'^2} \quad (22)$$

such that it satisfies the constraint of (23) and (24),

$$G_j + G'_j = G, \quad \forall G_j \geq 0, \quad G'_j \geq 0, \quad \forall j \in J, \quad (23)$$

$$0 \leq u'_j \leq U'_j, \quad \forall j \in J \quad (24)$$

The work further shows the optimal performance can be attained between CPU-GPU in terms of energy-load optimization by following condition 2 in (25),

$$\frac{\partial \mathcal{E}_j^{CPU}}{\partial G_j} = \frac{3C_j G_j^2}{u_j'^2}, \quad \forall j \in J \quad (25)$$

where \mathcal{E}_j^{CPU} defines the energy consumption of the CPU and is obtained as (26),

$$\mathcal{E}_j^{CPU} = \frac{C_j G_j^3}{u_j'^2} \quad (26)$$

and

$$\frac{\partial \mathcal{E}_j^{GPU}}{\partial G'_j} = \frac{3G_j G_j'^2}{u_j'^2}, \quad \forall j \in J, \quad (27)$$

where \mathcal{E}_j^{GPU} defines energy consumption of GPU and is obtained as described in (28),

$$\mathcal{E}_j^{GPU} = \frac{G_j G_j'^3}{u_j'^2} \quad (28)$$

Therefore, using the above equation, the optimal resource allocation is obtained through (29),

$$\frac{\partial \mathcal{E}_j^{CPU}}{\partial G_j} = \frac{\partial \mathcal{E}_j^{GPU}}{\partial G'_j}, \quad \forall j \in J, \quad (29)$$

with $G_j + G'_j = G$. The equation (22) requires maximizing the makespan of each computational node to achieve optimal performance as (22) represents a non-increasing function in $u'_j, \forall j \in J$; thus, it makes $u_j'^* = U'_j, \forall j \in J$, not dependent on workload partitioning. Therefore, using both condition 1 and condition 2, the optimal workload resource optimization can be obtained through (30),

$$G_j^* = \frac{\sqrt{C_j} G}{\sqrt{C_j} + \sqrt{G_j}}, \quad \text{and} \quad G_j'^* = \frac{\sqrt{G_j} G}{\sqrt{C_j} + \sqrt{G_j}}, \quad j \in J, \quad (30)$$

where C_j defines the computational factor of CPU and G_j defines the computational factor of GPU. Alongside, the optimal processor frequency-scaling of the CPU-GPU platform is obtained through (31),

$$f_j^* = \frac{G_j^*}{U'_j}, \quad f_j'^* = \frac{G_j'^*}{U'_j}, \quad j \in J, \quad (31)$$

where parameter U'_j defines the permissible maximum makespan of corresponding node j . The complete working of the proposed TFM-OTP model is explained in Algorithm 1.

3.5. Algorithm of proposed approach

The complete working process of task failure aware optimal task partitioning model is given in Algorithm 1. First, in step 2, the heterogeneous cloud platform is configured with the required number of physical machines (PMs) and virtual machines (VMs). For easiness, the work considered identical CPU-GPU configuration in all VMs. In step 3, the users start submitting the workload to a heterogeneous cloud platform with strict deadlines. Then, in step 4, the work partitions the data using (7) based on task dependencies levels and available CPUs and GPUs. In step 5, using (8) it computes the expected makespan, and in step 6, using (16) the model measures expected energy consumption for executing tasks in CPU-GPU. In step 7, using (22)

and meeting constraints (23) and (24) the work starts scheduling task X within workload G . In step 8, once all the tasks X with G are done the model measures the performance of the TFM-OTP scheduling model. The proposed model reduces task failures and is energy efficient by allocating to processors with minimal makespan aiding minimal energy consumption and better computational efficiency.

Algorithm 1. TFM-OTP

- Step 1. Start
- Step 2. Deploy heterogeneous CPU-GPU enabled cloud platform composed of a set of physical machines (PMs) and virtual machines (VMs) composed of CPU and GPUs.
- Step 3. User submits workload G to a heterogenous cloud platform with a deadline prerequisite.
- Step 4. The cloud platform first partitions the workload G according to task X dependencies level using (7).
- Step 5. The cloud platform then computes the approximate local makespan u'_j using (8).
- Step 6. The heterogenous cloud platform computes energy consumption \mathcal{E}_j^{cmp} of all VMs using (16).
- Step 7. The cloud platform schedules all the tasks X of workload G using (22) with meeting constraints (23) and (24) to reduce task failures with minimal makespan and energy consumption.
- Step 8. Measure the performance of the TFM-OTP scheduler in terms of makespan, energy consumption, and task failure energy overhead.
- Step 9. Stop

4. RESULTS AND DISCUSSION

This section studies the performance of TFM-OTP over existing workload scheduling, namely EMS [20]. The workload scheduling models are implemented using CloudSim. The energy consumption, makespan, and task failure energy overhead are metrics used for validating the performance of workload scheduling models. CyberShake and SIPHT are benchmark workloads that are used for experiment analysis which is available from [26].

4.1. Energy consumption

In this section, the energy efficiency of EMS and TFM-OTP is studied. Energy efficiency is measured concerning total energy consumed measured in watts for the execution of the scientific workload. Figure 1 shows the energy consumed for the execution of a SIPHT workload of a varying size where 30 represents a smaller workload and 1000 represents a larger workload. The result shows the TFM-OTP reduces energy usage by 39.35% in comparison with EMS for the execution of SIPHT workload. Similarly, Figure 2 shows the energy consumed for the execution of a CyberShake workload of varying size where 30 represents a smaller workload and 1000 represents a larger workload. The result shows the TFM-OTP reduces energy usage by 39.35% in comparison with EMS for execution of CyberShake workload. The energy efficiency improvement observed using TFM-OTP over EMS is due to the adoption of the energy optimization mechanism presented in (17) meeting performance constraints. However, EMS just optimizes the energy as per application reliability; thus, exhibits higher makespan and energy consumption which is observed in Figures 1 and 2.

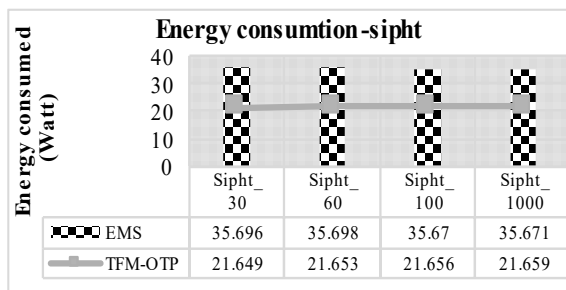


Figure 1. Energy consumption vs SIPHT workload size

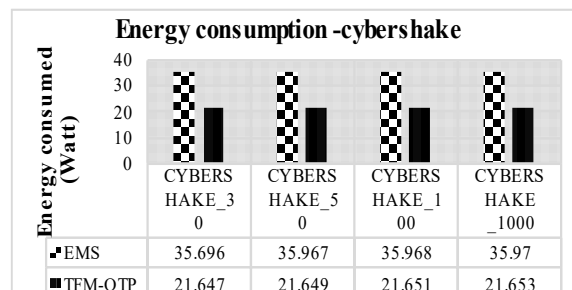


Figure 2. Energy consumption vs CyberShake workload size

4.2. Makespan

In this section, the makespan efficiency of EMS and TFM-OTP is studied. The makespan efficiency is measured concerning the total time taken measured in seconds for the execution of the scientific workload.

Figure 3 shows the makespan efficiency for the execution of a SIPHT workload of varying sizes. The result shows the TFM-OTP reduces makespan induced by 44.23% in comparison with EMS for the execution of SIPHT workload. Similarly, Table 2 shows the makespan efficiency for the execution of CyberShake workloads of varying sizes. The result shows the TFM-OTP reduces makespan induced by 97.33% in comparison with EMS for execution of CyberShake workload. The EMS schedules the task to VMs that minimize energy and meet task reliability; however, makespan minimization is not considered to lead to higher makespan at the cost of energy as seen in Figure 3 and Table 2. However, the makespan efficiency improvement achieved using TFM-OTP over EMS is because of the task being executed optimally in CPU-GPU pair as shown in (17); further, the performance efficiency with better load optimization meeting performance constraint is obtained using (22). Thus, a significant reduction of makespan in comparison with EMS can be observed in Figure 3 and Table 2.

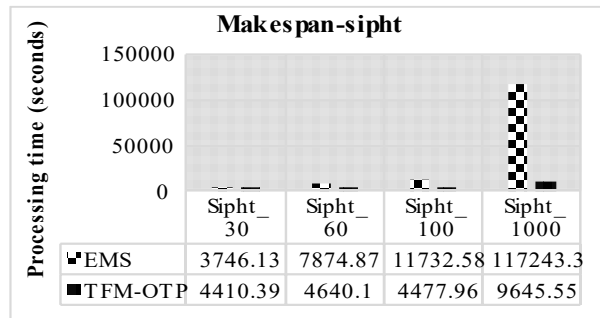


Figure 3. Makespan vs SIPHT workload size

Table 2. Makespan performance measured in seconds for CyberShake workflow

| Methodology | CyberShake_30 | CyberShake_50 | CyberShake_100 | CyberShake_1000 |
|-------------|---------------|---------------|----------------|-----------------|
| EMS | 6359.41 | 12380.99 | 24712.57 | 43685.27 |
| TFM-OTP | 262.73 | 283.68 | 323.35 | 1276.1 |

4.3. Task failure energy overhead

In this section, the task failure energy overhead performance of EMS and TFM-OTP is studied. The overhead is measured concerning the total energy consumed measured in watts for the execution of the failed task. Figure 4 shows the energy overhead for the execution of a SIPHT workload of varying size. The result shows the TFM-OTP reduces energy overhead by 69.14% in comparison with EMS for the execution of SIPHT workload. Similarly, Table 3 shows the energy overhead for the execution of CyberShake workloads of varying sizes. The result shows the TFM-OTP reduces energy overhead by 98.26% in comparison with EMS for execution of CyberShake workload. The task failure energy overhead reduction observed using TFM-OTP over EMS is due to the adoption of effective task partition at CPU-GPU pair-based resource availability meeting constraint defined in (17) and (22) meeting performance and load optimization constraint. However, EMS just optimizes the energy as per application reliability; thus, exhibits higher makespan and energy consumption which is observed in Figure 4 and Table 3.

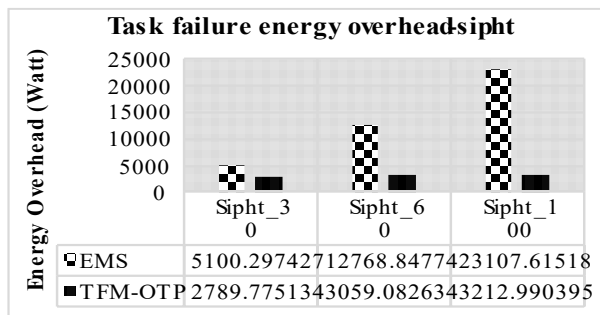


Figure 4. Average energy consumption vs SIPHT workload size

Table 3. Task failure energy overhead measured in watts for CyberShake workflow

| Methodology | CyberShake_30 | CyberShake_50 | CyberShake_100 |
|-------------|---------------|---------------|----------------|
| EMS | 6418.98 | 12720.34 | 26476.74 |
| TFM-OTP | 171.07 | 196.81 | 263.71 |

5. CONCLUSION

The study identified that most of the current methods to reduce energy leveraged DVFS; however, it affects the reliability of the computation platform; Especially, for executing workload applications the current method increases the failure rate. In addressing this paper introduces task partitioning into CPU-GPU pairs based on the local gradient computation process. Then DVFS-based energy consumption model for the execution of tasks in the CPU-GPU platform is presented; the model introduces an energy-performance tradeoff problem; further, the tradeoff model is extended to an energy-load optimization problem to provide reliability *i.e.*, reduce task failure energy overhead of heterogeneous CPU-GPU platform. Finally, an optimal scheduling scheme is presented. The experiment outcome shows TFA-OTP model reduces energy consumption by 39.35%, reduces makespan by 44.23%, and task failure energy overhead by 69.14% in comparison with EMS for SIPHT workload. Similarly, the experiment outcome shows TFA-OTP model reduces energy consumption by 39.35%, reduces makespan by 97.33%, and task failure energy overhead by 98.26% in comparison with EMS for CyberShake workload. The TFA-OTP is efficient in executing both large and small workloads with more energy and makespan efficiency meeting application reliability. However, in the future, the TFA-OTP will be tested considering different data-intensive scientific workloads.

REFERENCES

- [1] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, and A. El-Sayed, "Cost-effective algorithm for workflow scheduling in cloud computing under deadline constraint," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3765–3780, 2019, doi: 10.1007/s13369-018-3664-6.
- [2] X. Tang and Z. Fu, "CPU-GPU utilization aware energy-efficient scheduling algorithm on heterogeneous computing systems," *IEEE Access*, vol. 8, pp. 58948–58958, 2020, doi: 10.1109/ACCESS.2020.2982956.
- [3] M. Majewski, M. Pawlik, and M. Malawski, "Algorithms for scheduling scientific workflows on serverless architecture," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, May 2021, pp. 782–789, doi: 10.1109/CCGrid51090.2021.00095.
- [4] M. A. Rakrouki and N. Alharbe, "QoS-aware algorithm based on task flow scheduling in cloud computing environment," *Sensors*, vol. 22, no. 7, 2022, doi: 10.3390/s22072632.
- [5] A. Taghinezhad-Niar, S. Pashazadeh, and J. Taheri, "QoS-aware online scheduling of multiple workflows under task execution time uncertainty in clouds," *Cluster Computing*, vol. 25, no. 6, pp. 3767–3784, 2022, doi: 10.1007/s10586-022-03600-8.
- [6] B. Kocot, P. Czarnul, and J. Proficz, "Energy-aware scheduling for high-performance computing systems: A survey," *Energies*, vol. 16, no. 2, 2023, doi: 10.3390/en16020890.
- [7] H. R. Faragardi, M. R. Saleh Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: a budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020, doi: 10.1109/TPDS.2019.2961098.
- [8] R. Medara, R. S. Singh, and Amit, "Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization," *Simulation Modelling Practice and Theory*, vol. 110, Jul. 2021, doi: 10.1016/j.simpat.2021.102323.
- [9] M. Hussain, L. F. Wei, A. Rehman, F. Abbas, A. Hussain, and M. Ali, "Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers," *Future Generation Computer Systems*, vol. 132, pp. 211–222, 2022, doi: 10.1016/j.future.2022.02.018.
- [10] A. Albtoush, F. Yunus, K. Almi'ani, and N. M. M. Noor, "Structure-aware scheduling methods for scientific workflows in cloud," *Applied Sciences*, vol. 13, no. 3, 2023, doi: 10.3390/app13031980.
- [11] J. K. Konjaang and L. Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing," *Journal of Cloud Computing*, vol. 10, no. 1, 2021, doi: 10.1186/s13677-020-00219-1.
- [12] X. Tang *et al.*, "Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 9, pp. 2079–2092, 2022, doi: 10.1109/TPDS.2021.3134247.
- [13] X. Tang, "Reliability-aware cost-efficient scientific workflows scheduling strategy on multi-cloud systems," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2909–2919, 2022, doi: 10.1109/TCC.2021.3057422.
- [14] 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, pp. 860–875, 2022, doi: 10.1109/TSC.2019.2963382.
- [15] M. Kumar, A. Kishor, J. Abawajy, P. Agarwal, A. Singh, and A. Y. Zomaya, "ARPS: Sn autonomic resource provisioning and scheduling framework for cloud platforms," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, pp. 386–399, 2022, doi: 10.1109/TSUSC.2021.3110245.
- [16] M. C. Calzarossa, M. L. D. Vedova, L. Massari, G. Nebbione, and D. Tessera, "Multi-objective optimization of deadline and budget-aware workflow scheduling in uncertain clouds," *IEEE Access*, vol. 9, pp. 89891–89905, 2021, doi: 10.1109/ACCESS.2021.3091310.
- [17] S. Qin, D. Pi, Z. Shao, and Y. Xu, "A knowledge-based adaptive discrete water wave optimization for solving cloud workflow scheduling," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 200–216, 2023, doi: 10.1109/TCC.2021.3087642.
- [18] S. Qin, D. Pi, Z. Shao, Y. Xu, and Y. Chen, "Reliability-aware multi-objective memetic algorithm for workflow scheduling problem in multi-cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 4, pp. 1343–1361, 2023, doi: 10.1109/TPDS.2023.3245089.




- [19] X. Wang, J. Cao, and R. Buyya, "Adaptive cloud bundle provisioning and multi-workflow scheduling via coalition reinforcement learning," *IEEE Transactions on Computers*, vol. 72, no. 4, pp. 1041–1054, 2023, doi: 10.1109/TC.2022.3191733.
- [20] B. Hu, Z. Cao, and M. C. Zhou, "Energy-minimized scheduling of real-time parallel workflows on heterogeneous distributed computing systems," *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2766–2779, 2022, doi: 10.1109/TSC.2021.3054754.
- [21] N. Garg, Neeraj, M. Raj, I. Gupta, V. Kumar, and G. R. Sinha, "Energy-efficient scientific workflow scheduling algorithm in cloud environment," *Wireless Communications and Mobile Computing*, vol. 2022, 2022, doi: 10.1155/2022/1637614.
- [22] J. Kang and H. Yu, "GPGPU task scheduling technique for reducing the performance deviation of multiple GPGPU tasks in RPC-based GPU virtualization environments," *Symmetry*, vol. 13, no. 3, 2021, doi: 10.3390/sym13030508.
- [23] H. E. Zahaf, I. S. Sanudo Olmedo, J. Singh, N. Capodieci, and S. Faucou, "Contention-aware GPU partitioning and task-to-partition allocation for real-time workloads," in *ACM International Conference Proceeding Series*, 2021, pp. 226–236, doi: 10.1145/3453417.3453439.
- [24] A. Zou, J. Li, C. D. Gill, and X. Zhang, "RTGPU: real-time GPU scheduling of hard deadline parallel tasks with fine-grain utilization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 5, pp. 1450–1465, 2023, doi: 10.1109/TPDS.2023.3235439.
- [25] L. Wan, W. Zheng, and X. Yuan, "Efficient inter-device task scheduling schemes for multi-device co-processing of data-parallel kernels on heterogeneous systems," *IEEE Access*, vol. 9, pp. 59968–59978, 2021, doi: 10.1109/ACCESS.2021.3073955.
- [26] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, 2013, doi: 10.1016/j.future.2012.08.015.

BIOGRAPHIES OF AUTHORS



Divyaprabha Kabbal Narayana    received the B.E. degree in computer science from the University of Mysore, India, the M.Tech. degree in computer Science from the Visvesvaraya Technological University, Bangalore, India, and currently pursuing Ph.D. PES University Bangalore India. Her area of interest includes compilers and high-performance computation on homogeneous and heterogeneous platforms using machine learning and deep learning frameworks. She has authored or coauthored more than 10 publications. She can be contacted at email: divyaprabhamadhu@gmail.com.



Sudarshan Tekal Subramanyam Babu    is professor and dean of research at PES University, Bangalore India with 26 years of teaching experience. He Holds a PhD degree in computer networks and computer architecture from BITS, Pilani. He holds specialization in Heterogeneous computer architecture and computing, embedded systems and robotics, mobile networks and IoT. He is a senior member IEEE, member IET, member ACM, founder chairman of IEEE RAS Bangalore Chapter. He has more than 80+ publications in conference and 20+ journals. He also authored many book chapters. He can be contacted at email: sudarshan.tsb@gmail.com.