Reliable and efficient webserver management for task scheduling in edge-cloud platform

Sangeeta Sangani, Rudragoud Patil
Department of Computer Science and Engineering, K. L. S. Gogte Institute of Technology, Belgavi, Karnataka, India

Article Info

Article history:

Received Nov 15, 2022
Revised Feb 7, 2023
Accepted Feb 10, 2023

Keywords:

Cloud computing
Edge-computing
Quality of service
Reliability
Scheduling
Webserver
Workflow

ABSTRACT

The development in the field of cloud webserver management for the execution of the workflow and meeting the quality-of-service (QoS) prerequisites in a distributed cloud environment has been a challenging task. Though, internet of things (IoT) of work presented for the scheduling of the workflow in a heterogeneous cloud environment. Moreover, the rapid development in the field of cloud computing like edge-cloud computing creates new methods to schedule the workflow in a heterogenous cloud environment to process different tasks like IoT, event-driven applications, and different network applications. The current methods used for workflow scheduling have failed to provide better trade-offs to meet reliable performance with minimal delay. In this paper, a novel web server resource management framework is presented namely the reliable and efficient webserver management (REWM) framework for the edge-cloud environment. The experiment is conducted on complex bioinformatic workflows; the result shows the significant reduction of cost and energy by the proposed REWM in comparison with standard webserver management methodology.

Corresponding Author:

Sangeeta Sangani
Department of Computer Science and Engineering, K. L. S. Gogte Institute of Technology
Belgavi, Karnataka, India
Email: gitsangeeta@gmail.com

1. INTRODUCTION

The cloud webservice providers (CWP) take the tasks execution data in the form of a directed acyclic graph (DAG), which is known as a workflow. The scheduling of the workflow in the field of cloud has been researched for a long time [1]–[3]. Though, a major problem in workflow scheduling is providing better performance, reliability, energy efficiency, and fault tolerance in a large computing environment. Some frameworks automatically assign required resources, energy constraints, and performance for the tasks in the workflow for execution. Various researchers have proposed various methods and technologies which can reduce the consumption energy rate to conserve energy in the cloud environment and provide better performance in the cloud web server. Though, these methods need a large amount of communication cost between the web servers. Furthermore, these methods provide inadequate results and the consumption of energy cannot be decreased because of the high utilization of memory, and routing. The cost for communication between the webservers is high using the existing methods, as these methods depend on the traditional model which is not reasonable when applied to a hybrid cloud environment [3]–[5] for example, edge-cloud server as shown in Figure 1. This motivates the proposed work to design a workload scheduling model for edge-cloud platform that brings good tradeoffs between reducing and energy and cost and meets internet of things (IoT) workflow applications reliability requirement. In meeting research issues this paper present reliable and efficient web server resource management for workload execution in the hybrid cloud platform. Reliable and efficient webserver management (REWM) is designed to reduce energy consumption, reduce task failure, minimize delay, and provide high reliability. The research significance of REWM for task scheduling in the edge-cloud platform is as: i) the proposed work presents a novel workload management technique adopting an edge-cloud platform to provide efficiency and reliability; ii) the proposed REWM reduces task failures and provides fault-tolerance task offloading with minimal energy consumption; iii) the proposed REWM reduces energy consumption and cost by meeting efficiency and reliability constraints; and iv) the experiment outcome shows a significant reduction in energy and costs in comparison with existing workload management methodologies.

The paper is formatted in the given way. In section 2, literature survey on various existing methods for workload scheduling has been presented. In section 3, the methodology of the model is provided. In section 4, the results of the model using the epigenomics data set are presented and finally, in section 5, the conclusion and future work for the model has been given.

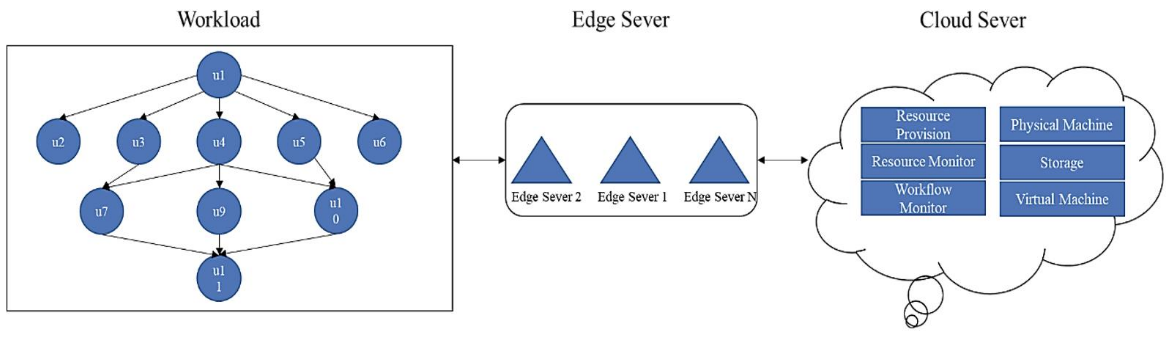


Figure 1. Heterogenous web server architecture for workload execution

2. LITERATURE SURVEY

In [3], they have proposed a method using the heterogenous-earliest-finish-time (HEFT), fuzzy dominance sort based heterogeneous earliest-finish-time (FDHEFT), to optimize the cost and makespan for the execution of the workloads in a heterogenous cloud. This method uses fuzzy dominance regulation for the execution of the workload. The experimentation performance has been evaluated using both the real-time and synthetic workload which has a better makespan and good cost trade-offs when contrasted with the traditional execution of the workload models. In [4], proposed a technique, days cash on hand (DCOH), for the hybrid cloud environment to optimize and reduce the cost and the deadline for the given task. The attained results give an outcome that the technique can efficiently reduce the trade-offs between the performance of the makespan and cost. Moreover, by reducing the consumption of energy, the cost of the execution of the workload can be reduced. In [5], they have proposed an energy-aware method to execute the DAG workloads having deadline constraints to reduce the consumption of energy in hybrid clouds. This model reduces the computational overhead of an energy-aware processor. In [6], a technique to reduce the cost and to attain better trade-off performance between energy and cost using the energy-aware scheduling method has been proposed. This technique comprises the given phases: slack resource optimization to save energy, idle virtual machine (VM) resource reuse policies, VM selection, and task merging. This technique attains good performance when compared with the existing workflow models. In [7], a model to optimize the cost and energy using the scheduling method to execute a large number of scientific data in IoT devices under a cloud network has been presented. This model improves performance, reduces energy consumption, and decreases the cost of execution of the task by a given deadline. In [8], a model to address the problem of reliability and to reduce the consumption of energy in the workload scheduling to provide better quality-of-service (QoS) requirements has been proposed. The experimental results of the model showed better efficiency and reliability for workload scheduling when compared with different models. Nowadays many algorithms like genetic algorithm, swarm optimization, and other algorithms are currently being used to solve the multi-objective workload scheduling (MOWS) issues in the cloud environment and to execute the real-time-workloads [9], [10] and also to optimize energy aware multi-objective function [11] presented energy minimized scheduling (EMS) [12]. Some other algorithms like reinforcement learning (RL) also have been used to solve the MOWS problems in the cloud network [13]–[15]. In [16], an improved fuzzy logic rule with GT to balance and control the load between the physical machines has been presented.

In [17], a Q-learning model to optimize the deadline and balance load for a given task having a weighted-objective function to schedule the workload has been presented. In [18], a technique, RADAR, to allocate resources to a given task in the cloud environment has been presented. This technique handles unpredicted failure and dynamic resource management considering the workload conditions. This technique decreases the cost required for execution, time, and service level agreements (SLA) violation when compared with the traditional techniques used for the execution of the workloads. In [19], they have proposed a model for the workflow which consists of composite tasks (cWFS). In this model, they have used nested-particle swarm optimization (N-PSO) method to execute the inner and outer population tasks. As this method is slow, a faster version of N-PSO has been used which executes the tasks by a given deadline. In [20], they have given an algorithm for the scheduling of the tasks named QL-HEFT which uses the Q-learning method and HEFT method to decrease the makespan during the execution of the task. This algorithm first ranks the given tasks on the given priority and then sorts them based on Q-learning and then gives an optimal resource utilization for each task so that the task can be executed by a given deadline. In [21], they have also proposed a scheduling method named as endpoint-communication contention-aware list-scheduling-heuristic (ELSH) which decreases the makespan of the workflow. In [22], they have presented an algorithm, DMWHDBS, which executes the task within the given deadline and is also cost-efficient. In this model, they have implemented a judgment mechanism that provides a success rate for the scheduling of the task in a multi-workflow environment. In [23], they have presented an allocation of resources technique for multi-cloud scheduling which executes the workflow, provides better performance, and reduces the cost of execution. In [24], they have designed a fault-tolerant workflow-scheduling model for the multi-cloud which reduces the cost during the execution and provides better reliability. They have also given a billing method for the resources utilized during the execution of the workflow. Finally, they have given an algorithm that reduces the cost, and time and also provides a solution for fault-tolerant workflow scheduling. The comparative study of two most recent workload scheduling is provided in Table 1. The proposed REWS work is focused to address limitation of existing model and are given in Table 1.

Table 1. Comparative study

	EMS [12], 2022	Reliability-aware cost-efficient scientific (RACES) [24], 2022	Proposed REWS
Heterogeneous computing	Yes	Yes	Yes
Edge-cloud	No	No	Yes
Workload type	Complex	Complex	Complex
Workload size	Small to large	Small to large	Small to large
QoS Metrics	Energy	Cost and time	Energy, Cost and Reliability
Optimization strategy	Heuristic	Weibull distribution	Heuristic
Planning	No	No	Yes
Reliability	Yes	Yes	Yes
Availability	Yes	No	Yes

3. METHOD

3.1. System and workflow execution on hybrid cloud environment

In this model, for the execution of the scientific workflow, a hybrid cloud environment has been considered. Suppose there are two sensor devices connected and both devices are running through the DAG application. The illustration of the workflow can be seen in Figure 1. The sensor devices are linked to an edge server which computes various operations and the edge server is linked to the cloud data center. The cloud data center comprises different hosts and n number of virtual machines. This model presumes that for the computation and communication process, there should be a stable connection between the sensor devices and the edge server. The server has the capability for the computation of scientific workflows and to execute the workflows within a given deadline.

3.2. Workload execution model for hybrid cloud environment

The execution of the workload is done either in the server or the cloud network when the sub-tasks are offloaded to the cloud. The delay induced to execute of the sub-task u_j in the server containing the processing element G_0 is defined using the (1). Further, the energy consumed to execute a given task of the workload in the edge server is represented using the (2).

$$M_j^m = \frac{\alpha_j}{G_0} \tag{1}$$

$$\mu_j^m = \varphi M_j^m \tag{2}$$

In (2), the energy consumed by the processing element for a given deadline is represented using φ. In the same way, the delay used to execute the offloading of the workload in the cloud network is calculated. The calculation of the capacity in the cloud is given using (3).

$$\mathbb{G} = \{G_1, G_2, G_3, \dots, G_n\} \tag{3}$$

The delay used to complete the execution of the workload comprises execution and communication delays. Hence, the total delay to execute the complete workload in the cloud is given using (4). Moreover, the delay used for the execution of the workload locally in the cloud or the edge server is calculated by the given using (5).

$$M_{jk}^{0} = \frac{\alpha_j}{G_k + M_{jk}^{s}} \tag{4}$$

$$M_{jk} = \frac{\alpha_j}{G_k + M_{jk}^{s}} \tag{5}$$

In (5), the value of $M_{jk}^{s} = 0$ when the value of $k=0$. The failure of the task in the sub-task of the workload u_k is calculated using the poisson distribution on the processing element t_k by the given (6). The efficiency of the processing element for the execution of the DAG workload is given using the (7) [19]. In (7), the sub-task index is defined using the y_{jk}. The sub-task y_{jk} can be described using (8).

$$S_{jk} = f^{-\omega_k \alpha_k / G_k} \tag{6}$$

$$S(H) = \prod_{s_j \in \mathbb{T}} \sum_{t_k \in \mathbb{T}} y_{jk} S_{jk} \tag{7}$$

$$y_{jk} = \begin{cases} 0, & \text{if a task } u_j \text{ is given to the processing element } t_k \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

In this model, the consumption of energy is reduced for the execution of the workload task H by decreasing the delay and increasing the efficiency of performance. The consumption of energy for the execution of the workload in the edge server is calculated using the (9). After this, the main issue can be denoted using the (10). In (10), the given constraint should be satisfied using the objective function to achieve an effective workload execution result. In (11), the constraint for the overall execution of delay in the workload G should be less when compared with the other delay bound. In (12), the delay bound describes the performance efficiency of the workload which must be greater than the specified performance efficiency bounds. The (13), specifies that each sub-task either can be executed either in the cloud network or in the edge server.

$$\mu_j = y_j \mu_j^{m} + \sum_{k=1}^{n} y_{jk} \mu_{jk}^{s} \tag{9}$$

$$\min \sum_{j=1}^{o} \mu_j \tag{10}$$

$$M(H) \le M_{preq} \tag{11}$$

$$S(H) \le S_{preq} \tag{12}$$

$$\sum_{k=0}^{n} y_{jk} = 1 \tag{13}$$

$$\tau(u_j) \ge \tau(u_l) + \sum_{k=0}^{n} y_{lk} M_{lk}, \quad \forall u_l \in prec(u_j) \tag{14}$$

$$\tau(u_j) + \sum_{k=0}^{n} y_{jk} M_{jk} \le \tau(u_l), \quad \forall u_l \in subseq(u_j) \tag{15}$$

$$y_{jk} \in \{0,1\} \tag{16}$$

The (14) and (15), it specifies the subsequent sub-tasks have to wait until the previous sub-task has been completely executed. In (14) and (15), the time for the execution of the sub-task u_j is initialized using the $\tau(u_j)$. This model's main aim is to reduce the delay in the execution of the task and to improve the performance efficiency which depends on some of the constraints given in (11) to (16). In this model, the resource which consumes less energy is used to execute the workload in a hybrid cloud environment. In the initial step, the sub-tasks are well-ordered in increasing order to generate a proper sequence set $\widehat{\mathbb{U}}$ for the sub-tasks. In the next step, the delay bounds and processing efficiency for each sub-task having different processing elements in the cloud network and edge server are obtained. Finally, the resources are allocated to the sub-task $\hat{u}_j$ by finding an appropriate processing element that consumes less energy overhead and also satisfies the bounds for the execution of the sub-task $\hat{u}_j$.

3.3. Task ordering webserver management

Figure 1 shows the existence of dependencies among preceding and subsequent tasks. As a result, after completion of the preceding task, only the webserver assign the resource to subsequent tasks. On the other side, if a task doesn't have any dependencies the webserver allocates resources in parallel. Take Figure 1 for example, the task u_1 is executed first and u_2 waits for the resource from the webserver to start execution, while in the meantime the task u_3 is executed in parallel to minimize the delay. As a result, the web server needs to decide the order in which it executes the workload so that the parallel efficiency can be maximized and delay can be reduced. In this work the task's ascendent ordering outcome S_v is used to measure tasks selectivity using the (17).

$$y_{jk} \in \{0,1\}$$
$$S_v(u_j) = \bar{M} + \max_{u_{l\in sub(u_j)}, t_j \in \mathbb{T}, t_m \in \mathbb{T}} \{f_{jl}(t_k, t_m) + S_v(u_l)\} \quad (17)$$

where, $\bar{M}$ defines the mean delay induced to execute the task on different webservers in graph H. The parameter $\bar{M}$ is computed using the (18). In this work, the delay induced for communicating the offloaded tasks outcomes toward the edge server. Therefore, the parameter $f_{jl}(t_k, t_m)$ defines the delay induced for communicating from the edge-server to t_m considering m=0rrr.

$$\bar{M} = \frac{\sum_{j=1}^{o} \sum_{k=0}^{n} \left(\frac{\omega_j}{G_k}\right)}{(n+1)o} \quad (18)$$

$$f_{jl}(t_k, t_m) = 0, \quad (19)$$

If m≠0 then

$$f_{jl}(t_k, t_m) = M_{lm}^{s} = \frac{\mu_l}{w_m}, \quad (20)$$

before starting workload execution all the tasks are arranged in a descendent structure by S_v. If tasks u_j and u_l are to be allocated and ascendant order assures $S_v(u_j) > S_v(u_l)$, in such cases the u_j might be having higher selectivity in comparison with u_l. Further, an important to be noted when u_j is about to be processed, it must wait till its preceding task is completed meeting the bounds defined in (14) and (15).

3.4. Reliable and efficient workload execution webserver management

The proposed work is focused on establishing the best reliable webserver server that reduces delay and assures fault-tolerance of workload H. Let $\widehat{\mathbb{U}}$ define the assignment order of $\mathbb{U}$ and the task that must be allocated is defined as $\hat{u}_j (\hat{u}_j \in \widehat{\mathbb{U}})$. The task set that must be processed is defined as $\{\hat{u}_1, \hat{u}_2, \hat{u}_3, \dots \hat{u}_{j-1}\}$ and tasks that must be allocated to the web server are defined as $\{\hat{u}_{j+1}, \hat{u}_{j+2}, \hat{u}_{j+3}, \dots \hat{u}_o\}$. In the work during allocating $\hat{u}_j$ the reliability (i.e., fault-tolerance) outcome considered must be $S(\hat{u}_j)$. Therefore, the present fault-tolerance of H is obtained through the (21).

$$S(\hat{u}_j, H) = \sum_{l=1}^{j-1} S_b(\hat{u}_l) * S(\hat{u}_j) \sum_{l=j+1}^{o} S_{vb}(\hat{u}_l) \quad (21)$$

where, $S(\hat{u}_j, H)$ represent the fault-tolerance outcomes of H when allocating $\hat{u}_j$. $S_b(\hat{u}_l)$ represent the fault-tolerance outcome experienced by the allocated $\hat{u}_l$ obtains. $S_{vb}(\hat{u}_l)$ defines the reliability outcome that unallocated task $\hat{u}_l$ obtains. The fault-tolerance outcome of a random web server is either equal or lesser than 1, follows bounds defined in (12), task $\hat{u}_j$ should satisfy the (22).

$$S(\hat{u}_j, H) \geq S_{preq} \quad (22)$$

Taking (13) into (14) and substituting the (23).

$$S(\hat{u}_j) \geq S_{preq}/\left(\sum_{l=1}^{j-1} S_b(\hat{u}_l) * \sum_{l=j+1}^{o} S_{vb}(\hat{u}_l)\right)$$

$$\geq S_{preq}/\left(\sum_{l=1}^{j-1} S_b(\hat{u}_l) * \sum_{l=j+1}^{o} S_{vc}(\hat{u}_l)\right) \quad (23)$$

where,

$$S(\hat{u}_j) = \sum_{t_k \in \mathbb{T}} x_{jk} S_{jk} \quad (24)$$

$S_{vc}(\hat{u}_j)$ depicts the upper limits of reliability outcomes of the task $\hat{u}_l$ can experience is given in the (25).

$$S_{vc}(\hat{u}_j) = \max_{t_k \in \mathbb{T}} \{S_{lk}\} \quad (25)$$

In the proposed algorithm, we adopt an effective soft-computing-based searching strategy that presumptuous every unallocated task must be allocated to either edge server or cloud webserver with the maximum fault-tolerance outcome, and later establish obtainable y_{jk} for assuring (23) in minimizing solution space size. Further, the work aimed to meet efficiency requirements in getting the required webserver with minimal time for workload execution. In this work, the fastest initialization time (FIT) and the recent completion time (RCT) are used for reducing the task's execution time and sustaining the workload delay prerequisites. First, the FIT of the incoming task U_{inc} on a different individual web server u_k is given using (26).

$$FIT(u_{inc}) = 0 \quad (26)$$

Then, we can get another task $u_j's$ FIT on a different web server t_k using the (27). The RCT of the departure task u_{dep} is given using (28). In meantime, this work assumes the other task $u_j's$ RCT on a different web server u_k which is given using (29). Then, this work obtains every task $u_j's$ minimized execution delay rather than the overall delay prerequisite of H which is given using (30).

$$FIT(u_j, t_j) = \max_{u_l \in prec(u_j), t_m \in \mathbb{T}} \left\{FIT(u_l, t_m) + \frac{\omega_l}{G_m + f_{lj}(t_m, t_k)}\right\} \quad (27)$$

$$RCT(u_{dep}, t_k) = M_{preq} \quad (28)$$

$$RCT(u_j, t_j) = \min_{u_l \in sub(u_j), t_m \in \mathbb{T}} \left\{RCT(u_l, t_m) - \frac{\omega_l}{G_m + f_{jl}(t_k, t_m)}\right\} \quad (29)$$

$$\sum_{k=0}^{n} y_{jk} \left(RCT(u_j, t_j) - FIT(u_j, t_j)\right) \geq M_{jk}^{d} = \frac{\omega_m}{G_k} \quad (30)$$

The proposed REWM methodology working is given in algorithm 1 which is focused on reducing delay and providing fault-tolerance assurance with high reliability meeting task QoS constraints of each task, and assures the constraint are bounded with minimal energy dissipation. The proposed methodology is designed considering the following three phases. First, use the task's ascendent ordering outcome S_v for generating $\hat{\mathbb{U}}$. Second, use (23) and (29) for obtaining the fault-tolerance and efficiency constraint of every individual task on different web servers in the edge-server or cloud web servers. Lastly, when allocating a task $\hat{u}_j$ establish a web server with the minimal energy-cost assuring bounds of $\hat{u}_j$. Hence, this model can reduce both the consumption of energy and cost when compared with the existing web server management for task scheduling which is discussed below in the results section.

Algorithm 1. Reliable and efficient webserver management (REWM)

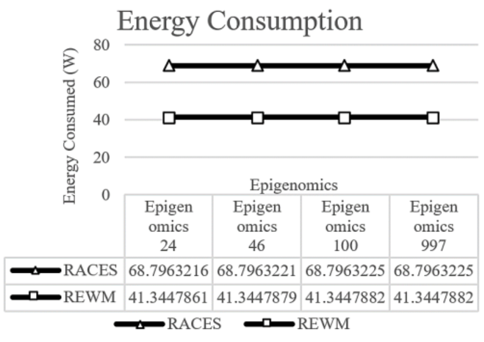
```
Step 1. Start
Step 2. Deploy edge-cloud platform with physical machines and virtual machines.
Step 3. The user submits workflow task with deadline requirement to edge-cloud resource provider.
Step 4. The resource provider first arranges the task according to its selectivity and arranges it in ascendent order
Step 5. The resource provider uses (23) and (29) for obtaining the fault-tolerance and efficiency constraint, respectively.
Step 6. The resource provider finds edge-server that reduces energy meeting deadline prerequisite using (10) and (23), respectively.
Step 7. If resource provider doesn't find any edge-server; then, the task is offloaded to cloud platform.
Step 8. The resource provider execute task in cloud platform that minimizing energy and meets efficiency and reliability constraint.
Step 9. Stop
```

4. RESULTS AND DISCUSSION

In this section, experiments have been conducted to evaluate the performance of the proposed model REWM over the existing reliability-aware cost-efficient scientific (RACES) workflows scheduling strategy on the multi-cloud systems model [24]. Experiments have been conducted on the epigenomics and SIPHT scientific workflow [25], [26]. The workflow has been discussed below in the next sub-section. Energy consumption, computation cost, and reliability are the three factors that have been considered to evaluate the performance of the model. The IoT-edge cloud server environment has been modeled using the SENSORIA simulator and the cloud environment is modeled using CloudSim and is combined through object-oriented programming language in building a hybrid cloud environment. The experiments have been conducted on an Intel i5 processor with NVIDIA graphics and RAM of 8 GB.

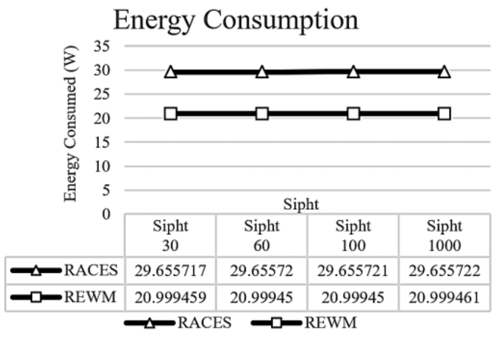
4.1. Energy consumption performance

In this section, the experiments have been conducted on both the epigenomics and SIPHT workflow by varying the size of the workload from 30 to 1,000 and the energy consumed for both the scientific workflow has been evaluated using the proposed REWM and the existing RACES model. From Figure 2 it can be seen that as the size of the workload of epigenomics workload increases the energy required for the execution also increases slightly. However, using the REWM model, a significant reduction in energy consumption can be seen in comparison to the existing RACES model. This shows that the REWM model achieves a scalable performance considering both the smaller workload and as well significantly large workload. Also, there is an average energy efficiency improvement of 16.63% which has been achieved by the REWM model in comparison with the RACES model. Further, from Figure 3 it can be seen that as the size of the workload of SIPHT workload increases the energy required for the execution also increases slightly. However, using the REWM model, a significant reduction in energy consumption can be seen in comparison to the existing RACES model. This shows that the REWM model achieves a scalable performance considering both the smaller workload and as well significantly large workload. Also, there is an average energy efficiency improvement of 6.009% which has been achieved by the REWM model in comparison with the RACES model.



	Epigenomics 24	Epigenomics 46	Epigenomics 100	Epigenomics 997
RACES	68.7963216	68.7963221	68.7963225	68.7963225
REWM	41.3447861	41.3447879	41.3447882	41.3447882

Figure 2. Energy consumption for epigenomics with different workload sizes



	Sipht 30	Sipht 60	Sipht 100	Sipht 1000
RACES	29.655717	29.65572	29.655721	29.655722
REWM	20.999459	20.99945	20.99945	20.999461

Figure 3. Energy consumption for the SIPHT with different workload sizes

4.2. Computational cost

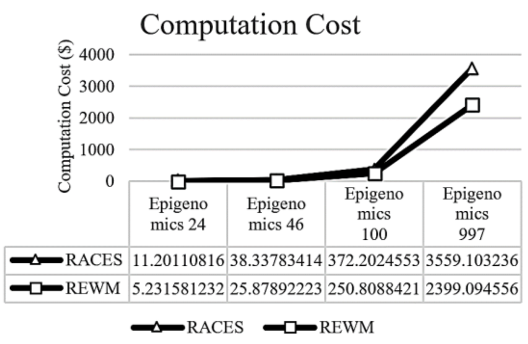
In this section, the experiments have been conducted on both the epigenomics and SIPHT workflow by varying the size of the workload from 30 to 1,000 and the cost consumption for both the scientific workflow has been evaluated using the proposed REWM and the existing RACES model. The cost is measured based on total time spent on respective server type on Azure cloud as defined using (31).

$$total\ cost = X * T \tag{31}$$

where X defines the total cost (measure in dollars) per second on respective instance type. More detail can be obtained from [24], [27]. From Figure 4 it can be seen that as the size of the workload of epigenomics workload increases the cost required for the execution also increases slightly. However, using the REWM model, a significant reduction in the computational cost can be seen in comparison to the existing RACES model. This shows that the REWM model achieves a scalable performance and reduces the computational cost considering both the smaller workload and as well significantly large workload. Also, there is an average energy efficiency improvement of 4.67% which has been achieved by the REWM model in comparison with the RACES model. Further, from Figure 5 it can be seen that as the size of the workload of SIPHT workload increases the computational cost required for the execution also increases slightly. However, using the REWM model, a significant reduction in the computational cost can be seen in comparison to the existing RACES model. This shows that the REWM model achieves a scalable performance and reduces the cost considering both the smaller workload and as well significantly large workload. Also, there is an average energy efficiency improvement of 6.44% which has been achieved by the REWM model in comparison with the RACES model.

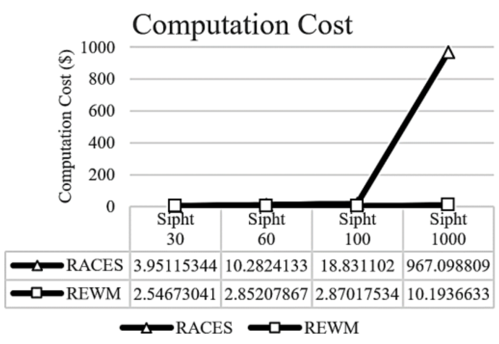
4.3. Reliability

In this section, the experiments have been conducted on both the Epigenomics and SIPHT workflow by varying the size of the workload from 30 to 1,000 and the reliability of the model for both the scientific workflow has been evaluated using the proposed REWM and the existing RACES model. From Figure 6 it can be seen that as the size of the workload of epigenomics workload increases the reliability of the REWM model increases in comparison with the existing RACES model. This shows that the REWM model achieves a scalable performance and reduces the computational cost and is highly reliable considering both the smaller workload and as well significantly large workload. Also, there is an average energy efficiency improvement of 0.065% which has been achieved by the REWM model in comparison with the RACES model. Further, From Figure 7 it can be seen that as the size of the workload of SIPHT workload increases the reliability of the REWM model increases in comparison with the existing RACES model. This shows that the REWM model achieves a scalable performance and reduces the computational cost and is highly reliable considering both the smaller workload and as well significantly large workload. Also, there is an average energy efficiency improvement of 0.115% which has been achieved by the REWM model in comparison with the RACES model.



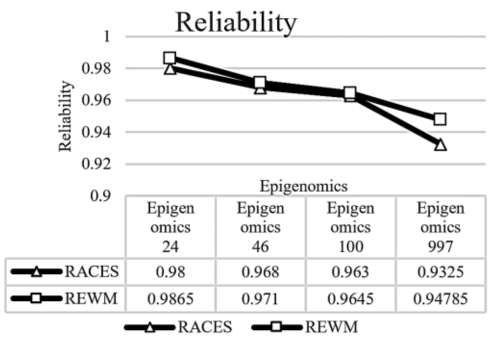
	Epigenomics 24	Epigenomics 46	Epigenomics 100	Epigenomics 997
RACES	11.20110816	38.33783414	372.2024553	3559.103236
REWM	5.231581232	25.87892223	250.8088421	2399.094556

Figure 4. Computation cost for epigenomics with different workload sizes



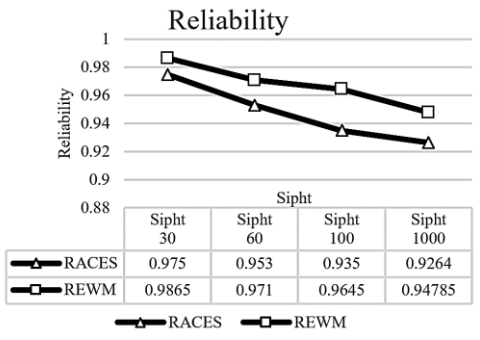
	Sipht 30	Sipht 60	Sipht 100	Sipht 1000
RACES	3.95115344	10.2824133	18.831102	967.098809
REWM	2.54673041	2.85207867	2.87017534	10.1936633

Figure 5. Computation cost for the SIPHT with different workload sizes



	Epigenomics 24	Epigenomics 46	Epigenomics 100	Epigenomics 997
RACES	0.98	0.968	0.963	0.9325
REWM	0.9865	0.971	0.9645	0.94785

Figure 6. Reliability for Epigenomics with different workload sizes



	Sipht 30	Sipht 60	Sipht 100	Sipht 1000
RACES	0.975	0.953	0.935	0.9264
REWM	0.9865	0.971	0.9645	0.94785

Figure 7. Reliability for the SIPHT with different workload sizes

5. CONCLUSION

In this paper, we have gone through various research presented on the scheduling of the workflow and came across different algorithms used to reduce energy consumption and cost in a cloud environment and also to increase reliability. In various studies, it could be seen that less amount of work has been done on workflow scheduling problems to reduce the cost and energy consumption in a heterogeneous cloud network. The present models for workload scheduling have failed to bring out a good trade-off when meeting the energy constraint and task deadline. In this model, we have presented an efficient method that can provide good trade-offs to meet the energy constraint and task deadline in an edge-cloud environment for provisioning complex IoT workflow. The REWM computes the delay for executing in the edge server and the task failure rate for IoT workflow execution. Then, the benefit (i.e., performance efficiencies) of minimizing delay and failure rate by offloading execution on the cloud platform is estimated. Finally, the energy is optimized to meet task delay and processing efficiency requirements. In this way, the REWM achieves much superior throughput, energy efficiency, and cost reduction for executing workflow on a heterogeneous platform and an increase in reliability when compared with RACES and EMS which is proven through experimental study for provisioning IoT workflows.

REFERENCES

[1] D. Poola, M. A. Salehi, K. Ramamohanarao, and R. Buyya, “A taxonomy and survey of fault-tolerant workflow management systems in cloud and distributed computing environments,” in *Software Architecture for Big Data and the Cloud*, Elsevier, 2017, pp. 285–320.
[2] R. Khorsand, F. Safi-Esfahani, N. Nematbakhsh, and M. Mohsenzade, “Taxonomy of workflow partitioning problems and methods in distributed environments,” *Journal of Systems and Software*, vol. 132, pp. 253–271, Oct. 2017, doi: 10.1016/j.jss.2017.05.017.
[3] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, “Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT,” *Future Generation Computer Systems*, vol. 93, pp. 278–289, Apr. 2019, doi: 10.1016/j.future.2018.10.046.
[4] J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, and M. Chen, “Cost and makespan-aware workflow scheduling in hybrid clouds,” *Journal of Systems Architecture*, vol. 100, Nov. 2019, doi: 10.1016/j.sysarc.2019.08.004.
[5] G. Xie, G. Zeng, R. Li, and K. Li, “Energy-aware processor merging algorithms for deadline constrained parallel applications in heterogeneous cloud computing,” *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 62–75, Apr. 2017, doi: 10.1109/TSUSC.2017.2705183.
[6] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, “Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds,” *IEEE Transactions on Services Computing*, vol. 11, no. 4, pp. 713–726, Jul. 2018, doi: 10.1109/TSC.2015.2466545.
[7] Y. Wen, Z. Wang, Y. Zhang, J. Liu, B. Cao, and J. Chen, “Energy and cost aware scheduling with batch processing for instance-intensive IoT workflows in clouds,” *Future Generation Computer Systems*, vol. 101, pp. 39–50, Dec. 2019, doi: 10.1016/j.future.2019.05.046.
[8] R. Garg, M. Mittal, and L. H. Son, “Reliability and energy efficient workflow scheduling in cloud environment,” *Cluster Computing*, vol. 22, no. 4, pp. 1283–1297, Dec. 2019, doi: 10.1007/s10586-019-02911-7.
[9] L. Chunlin, T. Jianhang, and L. Youlong, “Hybrid cloud adaptive scheduling strategy for heterogeneous workloads,” *Journal of Grid Computing*, vol. 17, no. 3, pp. 419–446, Sep. 2019, doi: 10.1007/s10723-019-09481-3.
[10] C. Li, J. Tang, and Y. Luo, “Cost-aware scheduling for ensuring software performance and reliability under heterogeneous workloads of hybrid cloud,” *Automated Software Engineering*, vol. 26, no. 1, pp. 125–159, Mar. 2019, doi: 10.1007/s10515-019-00252-8.
[11] M. Sardaraz and M. Tahir, “A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing,” *International Journal of Distributed Sensor Networks*, vol. 16, no. 8, Aug. 2020, doi: 10.1177/1550147720949142.
[12] 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*, vol. 15, no. 5, pp. 2766–2779, Sep. 2022, doi: 10.1109/TSC.2021.3054754.
[13] D. P. Bertsekas, “Feature-based aggregation and deep reinforcement learning: a survey and some new implementations,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 1–31, Jan. 2019, doi: 10.1109/JAS.2018.7511249.
[14] L. Xue, C. Sun, D. Wunsch, Y. Zhou, and F. Yu, “An adaptive strategy via reinforcement learning for the prisoner dilemma game,” *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 301–310, Jan. 2018, doi: 10.1109/JAS.2017.7510466.
[15] H. Wang, T. Huang, X. Liao, H. Abu-Rub, and G. Chen, “Reinforcement learning for constrained energy trading games with incomplete information,” *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3404–3416, Oct. 2017, doi: 10.1109/TCYB.2016.2539300.
[16] E. Iranpour and S. Sharifian, “A distributed load balancing and admission control algorithm based on fuzzy type-2 and game theory for large-scale SaaS cloud architectures,” *Future Generation Computer Systems*, vol. 86, pp. 81–98, Sep. 2018, doi: 10.1016/j.future.2018.03.045.
[17] W. Jiahao, P. Zhiping, C. Delong, L. Qirui, and H. Jieguang, “A multi-object optimization cloud workflow scheduling algorithm based on reinforcement learning,” in *Intelligent Computing Theories and Application*, 2018, pp. 550–559.
[18] S. S. Gill, I. Chana, M. Singh, and R. Buyya, “RADAR: Self-configuring and self-healing in resource management for enhancing quality of cloud services,” *Concurrency and Computation: Practice and Experience*, vol. 31, no. 1, Jan. 2019, doi: 10.1002/cpe.4834.
[19] 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, pp. 1074–1088, Mar. 2022, doi: 10.1109/TSC.2020.2975774.
[20] Z. Tong, X. Deng, H. Chen, J. Mei, and H. Liu, “QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment,” *Neural Computing and Applications*, vol. 32, no. 10, pp. 5553–5570, May 2020, doi: 10.1007/s00521-019-04118-8.
[21] 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, pp. 1137–1150, Apr. 2022, doi: 10.1109/TASE.2020.3046673.

[22] 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, pp. 4939–4949, Dec. 2021, doi: 10.1109/JSYST.2021.3087527.
[23] 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, Mar. 2022, doi: 10.1109/TSC.2019.2963382.
[24] 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, Oct. 2022, doi: 10.1109/TCC.2021.3057422.
[25] Pegasus, "Epigenomics," *Workflow gallery*. https://pegasus.isi.edu/workflow_gallery/gallery/epigenomics/index.php (accessed Feb. 06, 2023).
[26] Pegasus, "Sipht," *Workflow gallery*. https://pegasus.isi.edu/workflow_gallery/gallery/sipht/index.php (accessed Feb. 06, 2023).
[27] Azure, "Cloud computing services: microsoft azure, cloud computing services," *Microsoft Azure*. https://azure.microsoft.com./ (accessed Feb. 06, 2023).

BIOGRAPHIES OF AUTHORS

Sangeeta Sangani was born in 1982, in Karnataka. She received her B.E. Degree in Computer Science and Engineering from VTU University, Belagavi in 2005. Completed her post-graduation, M. Tech (CSE) in 2008 from VTU Belagavi. And currently pursuing a Ph.D. from VTU Belagavi. She has a total experience of 14 years in teaching in Technical Institutes like R.V. College of Engineering, Basveshwara Engineering College Bagalkot, and S. G. Balekundri Institute of Technology, Belagavi. And now since 2010 working as an Assistant Professor in KLS, Gogte Institute of Technology, Belagavi. In her total career, she has guided more than 10 UG projects and 5 PG projects and published over 7 papers in International Journals and Conferences. She can be contacted at email: gitsangeeta@gmail.com.

Rudragoud Patil currently working as an Associate Professor, at Dept of CSE, KLS Gogte Institute of Technology, Belagavi. He has 12 years of Teaching Experience at professional institutes across Karnataka. He published over 13 papers in International Journals, Book Chapters, and Conferences of High Repute. His subjects of interest include cloud computing, distributed computing, machine learning, and network security. He can be contacted at email: rspatil@git.edu.