

Efficient algorithm for replanning web service composition

Kavita D. Hanabaratti, Rudragoud Patil

Department of Computer Science and Engineering, K. L. S Gogte Institute of Technology, Belagavi, India

Article Info

Article history:

Received Oct 31, 2022

Revised Mar 23, 2023

Accepted Mar 24, 2023

Keywords:

Efficient re-planning

Montage-workflow

Reliable

Web-service-composition

Workload execution

ABSTRACT

Web-service-composition (WSC) workload execution inside a hybrid cloud environment is challenging. A dynamic approach for allocating resources to various tasks, as well as associated sub-tasks having a satisfactory quality-of-service (QoS) requirement, is necessary for the present real-time demand. As a result of focusing primarily on decreasing processing time as well as cost, current approaches improve latency as well as energy while executing a given workload. This study introduces an efficient re-planning (ERP) algorithm for running many scientific workloads inside a heterogeneous cloud environment, which is designed to address some of the shortcomings of previous approaches. With a changing workload, this study details a technique to improve the WSC's availability as well as robustness. The workload's processing energy requirements are reduced as a result. The montage-workflow has been used to validate the research findings. Comparison with the current heterogeneous earliest finish time (QL-HEFT) algorithm demonstrates that its ERP-WSC approach is much more efficient and reliable.

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



Corresponding Author:

Kavita D. Hanabaratti

Department of Computer Science and Engineering, K.L.S Gogte Institute of Technology

Belagavi, India

Email: kdhanabaratti@git.edu

1. INTRODUCTION

Processes based on directed-acyclic-graphs (DAGs) have been widely utilized for handling massive datasets that include complex scientific workflows comprising multiple interdependent tasks. The cloud supplies its customers with several virtual computational nodes, which they use to execute the workflow. Workflow computation can be distributed among a configurable set of cloud nodes such that the customer can take advantage of this functionality [1]. Customers often receive nodes following the terms of their service-level-agreements (SLAs). Service-level-agreements explain the requirements stipulated amongst the service operator as well as the customers to achieve the expected level of quality-of-service (QoS). Customers must pay a membership fee that is determined by the level of service level agreement and quality of service they demand. Not only that, but in DAG applications, neither [2] nor [3] have they addressed any strategy for decreasing energy consumption during a specific operation. Clouds are considered heterogeneous if platforms offer sufficient storage space for effective computation of workloads and support distributed computing, such is the case with cloud-computing [4], [5].

Heterogeneous cloud environments necessitate the adoption of much more expensive as well as power-intensive approaches [6]. The DAG processes as well as associated task dependencies are taken into account when deciding where in the cloud to run the tasks [7]-[10]. Several strategies have already been presented to simultaneously lower energy consumption as well as meet the SLA requirement for a specific work [11]-[13]. Moreover, when a workload requires a certain QoS, the cloud platform's scheduling is notified to look for a web-service as well as its parameters that meet those requirements. The scheduling algorithm has the option of making either a static or a dynamic static resource allocation to the workload. Yet, the workload's

execution needs to be finished within a certain time while using less energy than usual and according to green computing standards. Minimizing power usage was a primary goal of the conventional WCS approach. However, there is no guarantee that these techniques will work reliably. Hence, an efficient re-planning (ERP) method for enhancing dependability underneath a dynamic-workload was therefore developed in a heterogeneous cloud setting in this work, to resolve the issue of reliability within the context of web-based workload composition. The following is an outline of the various contributions made by the proposed work:

- The proposed approach is used for carrying out the web-service-composition tasks automatically.
- The presented approach solves the limitations of a particular service to fulfill complicated customer demands.
- The proposed approach provides a searching technique for planning the workload of web services to continuously update their availability.
- The proposed approach enhances the quality-of-service metric concerning its reliability and availability.

The following paper comprises the following section. In section 2, a literature survey on the existing works has been discussed. Further, in section 3, an ERP algorithm has been proposed for web-service-composition. Based on the algorithm, the results have been evaluated and compared with existing work in section 4. To summarize the complete work, in section 5, the conclusion and future work for the proposed ERP algorithm have been given.

2. LITERATURE SURVEY

In this section, we have examined a variety of recent strategies for the composition of workloads using resources that utilize the cloud as well as heterogeneous computing environments. The difficulties of using several cloud providers are addressed in [14], which proposes a solution. This approach begins by constructing a fault-tolerant-workflow scheduler system, that, among other benefits, contributes to a cost reduction as well as an improvement within execution dependability. Then, they establish the cost-effectiveness of a DAG task and present an FCWS method that reduces both the time and money spent on assuring reliability without compromising performance. The ideas, protocols, as well as approaches for cloud computing to minimize energy use, are laid forth in [15]. This paper discusses the demands and requirements of cloud customers. The paper additionally provides a comprehensive analysis of the energy efficiency, dependability, cost, as well as efficiency of the virtual machines (VM) placed in cloud data centers.

Xie *et al.* [16], they find a time-scheduling method that takes into account only the workflow's deadlines as well as the total scheduling duration. The goal of the total scheduling method is to maximize performance while minimizing the planning duration. The method selects the best virtual machine for each workflow based on its urgency and the time remaining until the workflow's completion. Khoramnejad *et al.* [17], a methodology for multimedia cloud computing task scheduling, as well as cache management, is presented with the goals of lowering operational expenses and increasing productivity and responsiveness. This approach combines a variety of characteristics, including VM processing speed, pre-fetcher utilization, as well as the frequency at which user requests arrive, to improve efficiency in terms of both expense and response duration. The findings demonstrate that somehow the expenses associated with data transfer rise in lockstep with the amount of data being sent.

A strategy for scheduling tasks in privatized clouds has indeed been presented in [18], to ensure that the workload makes efficient use of the available resources. Using a neural-network, this method guarantees that the workload's activities are completed by the user-specified deadline. They concluded that by taking this method, the price, as well as response duration inside the specific cluster, may be decreased, leading to improved QoS for the customer inside the privatized cloud. Xie *et al.* [19], an EPM method is described for choosing the best CPU for a given cloud computing function, then powering it down afterward to conserve energy. To reduce the time and effort required to calculate the EPM method, a new algorithm called QEPM has already been developed. The findings indicate that as contrasted to current methods, both methods aid in significantly reducing energy consumption.

Common European Asylum System (CEAS), developed in [20], is an efficient scheduler method for cutting down on cloud computing's overhead and power usage. There are 5 stages to the method. The efficiency of every procedure was evaluated by measuring its computation time. Current web service deployment struggles the most when faced with dynamically changing resource needs, delivering poor outcomes [21] and failing to provide effective trade-offs among lowering energy usage as well as meeting performance prerequisites [22], [23]. To execute workload applications with constantly fluctuating QoS inside the cloud, high quality of service with service level agreement assurance is required, and this can only be achieved through an efficient replanning technique.

3. EFFICIENT-REPLANNING ALGORITHM FOR WEB SERVICE COMPOSITION TO SCHEDULE THE WORKLOADS IN A HETEROGENOUS CLOUD ENVIRONMENT

In this section, first, the energy consumption methodology for the execution of the workload has been discussed. Further, the scheduling of the tasks in the web service composition (WSC) has been explained. Finally, an efficient-replanning algorithm has been proposed for providing better reliability.

3.1. Energy consumption methodology during the execution of the workload

A significant number of variable physical devices I compose the sophisticated heterogeneous-cloud infrastructure, which is shown mathematically as shown in (1).

$$I = \{I_1, I_2, I_3, \dots, I_o\}, \tag{1}$$

In (1) the o is used for representing the physical machine present in the heterogenous-cloud infrastructure. For every physical machine $I_l \in I$, which can be represented utilizing the given in (2).

$$I_l = \{st_l, n_l, q_l^\uparrow, o_l, (g_l, w_l), V_l\}, \tag{2}$$

In (2), st_l is used for representing the storage-size, n_l is used for representing how much space the memory can hold, q_l^\uparrow is used for representing a very high state of energy, o_l is used for representing the maximum amount of bandwidth available, (g_l, w_l) is used for representing the range for frequency as well as voltage. Further, the V_l is used for representing the virtual machines present inside the I . The (g_l, w_l) can be denoted using the given (3).

$$(g_l, w_l) = \{(g_l^1, w_l^1), (g_l^2, w_l^2), \dots, (g_l^\uparrow, w_l^\uparrow)\} \tag{3}$$

Further, the V_l can be represented by the given in (4).

$$V_l = \{v_{l,1}, v_{l,2}, \dots, v_{l,|V_k|}\} \tag{4}$$

Furthermore, the virtual machine inside the I is represented by the given (5).

$$v_{l,m} = \{g_{l,m}, n_{l,m}, st_{l,m}\} \tag{5}$$

In (5) $g_{l,m}$ is used for portraying the range of frequency of each virtual machine, $n_{l,m}$ is used for portraying the space of memory it can hold inside the virtual machine and $st_{l,m}$ is used for portraying storage-size inside the virtual machine. Furthermore, the hardware can be partitioned into virtual computers and ported across different hosts. The VMs could share hardware resources and move freely among many physical servers. Assume that the highest amount of energy that a specific physical-machine i_l can handle is q_l^\uparrow , as well as set the static consumption of energy at t_l . In (6) represents the energy used to run the workload just on the physical machine i_l .

$$I_l = t_l * q_l^\uparrow * z_l^u + \frac{(1-t_l)*q_l^\uparrow}{(g_l^\uparrow)^3} * (g_l)^3, \tag{6}$$

In (6), $z_l^u \in \{1,0\}$ is used for representing whether or not the i_l is non-active or active for the time u . At a given instant u , g_l represents the CPU speed. Highest CPU speed at time u is denoted by the symbol g_l^\uparrow . In (7) could be used to determine the energy consumed by a specific physical-machine o across a range of times based upon these characteristics.

$$\mathcal{E} = \sum_{j=1}^o \int_{xt}^{yt} \left(t_l * q_l^\uparrow * z_l^u + \frac{(1-t_l)*q_l^\uparrow}{(g_l^\uparrow)^3} * (g_l)^3 \right) dt, \tag{7}$$

Further, in (7), all the parameters are static except z_l^u as well as the g_l which fluctuates with time, due to which it is known as a time-dependent parameter.

3.2. Scheduling the tasks of the workload in WSC

Consider the mapping relationship that exists among the $v_{l,m}$ virtual machine and various u_k^j tasks that run on the i_l physical machine be defined by $y_{k,lm}^j$. Whenever the tasks u_k^j is mapped to the $v_{l,m}$ the virtual machine, then the $y_{k,lm}^j$ tends to 0. Further, when the tasks u_k^j is not mapped to the $v_{l,m}$ the virtual machine, then the $y_{k,lm}^j$ tends to 1. This can be represented using (8).

$$y_{k,lm}^j = \begin{cases} 0, & \text{if } u_k^j \text{ is not mapped to } v_{l,m}, \\ 1, & \text{otherwise.} \end{cases} \quad (8)$$

To solve the issue of resource allocation restrictions, data dependencies across tasks with varying processes and workloads must be resolved. This restriction can be solved using (9).

$$gu_{q,lm}^j + uu_{qk}^j \leq st_{k,lm}^j, \forall f_{qk}^j \in F_j \quad (9)$$

In (9), $gu_{q,lm}^j$ is used for representing the total time consumed for the execution of the workload, uu_{qk}^j is used for representing the transmission time between the tasks u_q as well as u_j . Once the task has been executed and the virtual machine's time x_j has completed, the necessary execution time can be calculated utilizing the given in (10).

$$gu_j = \max_{u_k^j \in U_j} \{gu_{k,lm}^j\}. \quad (10)$$

The tasks should complete the execution successfully under the deadline constraint after the necessary resources have been made available. The provided equation is utilized here for this purpose.

$$gu_j \leq e_j, \forall x_j \in X. \quad (11)$$

As the actual machine has fewer resources available, the use of virtual machines is constrained. Because of this issue, providing the necessary resources is a challenging process. Hence, the constraints have to be defined. By using the given (12) and (13), we can define the constraints.

$$g_l^\uparrow - \sum_{m=1}^{|V_l|} g_{l,m} \geq 0, \forall i_l \in I; \quad (12)$$

$$n_l - \sum_{m=1}^{|V_l|} n_{l,m} \geq 0, \forall i_l \in I. \quad (13)$$

To reduce the completion time for the execution of the scientific workload, this approach meets well all criteria indicated by (9), (11), (12), as well as (13), which require a decrease in energy usage during resource scheduling. The following mathematical expression can be used to represent this (14).

$$\text{Min } \sum_{l=1}^o \int_{xt}^{yt} \left(t_l * q_l^\uparrow * z_l^u + \frac{(1-t_l) * q_l^\uparrow}{(g_l^\uparrow)^3} * (g_l^e)^3 \right) dt. \quad (14)$$

In (14), o is used for representing the capacity of the physical machine, xt represents the beginning of the task being performed and yt represents the completion of the task. Furthermore, from (15) allows for an effective distribution of the available resources under this scenario.

$$\text{Max } \left(\sum_{j=1}^n \sum_{k=1}^{|U_j|} cpu_k^j * \mathcal{T}_k^j \right) / \left(\sum_{l=1}^o g_l^\uparrow * \mathcal{A}_l \right), \quad (15)$$

As shown in (15), n is used for representing the workload size, $|U_j|$ is used for representing the task-size x_j . The required frequency for the CPU is defined by cpu_k^j . \mathcal{T}_k^j is used for representing the total amount of time it takes to complete the workload's tasks, o is used for representing the physical machine, \mathcal{A}_l is used for representing the time that the physical machine has been active.

3.3. Efficient-replanning algorithm for WSC in the dynamic real-time environment

In this work, the task is scheduled using the min-max function defined in (14) and (15), respectively. However, finding resources using the above metrics induces certain constraints such as task dependency, the deadline prerequisite (i.e., application reliability prerequisite), and resource availability. Thus, scheduling tasks using the min-max trade-offs function is an NP-hard problem. In this work, a heuristic mechanism is designed to solve the trade-offs problem and obtain the optimal solution for the execution of a dynamic workload with reliability (i.e., deadline) prerequisite with time complexity. In the existing model, the resource provisioner takes input i.e., workload task, and schedules only a few tasks to the virtual machines, and the remaining task will be in waiting since the task have dependencies. Then with the arrival of a new workload, the ready-to-be-executed task fails to obtain the resource. As a result, task reliability is affected and resources are wasted. Thus, an efficient replanning algorithm is needed to optimize the flow of workload execution with the arrival of a new workload. So, in this process, before arranging the execution of the workflow, it is vital to plan the sequence of how the tasks are to be carried out. In this work, we describe an efficient re-planning technique by first defining tasks based on the most recent beginning period, ru_k^j , of each workload. These beginning durations are then utilized to order the execution of workload tasks. Every task u_k^j has a start duration represented as ru_k^j during which it has to be initiated; if this fails, the workload x_j completing time gu_k^j , may be affected, causing it to fail its deadline, thus, impacting its reliability. The start duration of the task can be represented by (16).

$$r(u_k^j) = \begin{cases} e_j - du_k^j, & \text{if } succ(u_k^j) = \emptyset \\ \min_{u_s^j \in (u_k^j)} \{(ru_s^j) - uu_{ks}^j - du_k^j\}, & \text{otherwise.} \end{cases} \quad (16)$$

In (16), $succ(u_k^j)$ is used to represent the complete task of the workload which also includes the u_k^j tasks that are in the queue. Hence, the recent duration rgu_k^j for the given task u_k^j can be evaluated from the given in (17).

$$rgu_k^j = ru_k^j + du_k^j. \quad (17)$$

Since each workload task establishes its own unique preferred bounds, many VMs may sit idle while their workload tasks run. Moreover, no data dependencies exist across various workloads. So, efficient planning of tasks across various workloads simultaneously would considerably contribute to lowering the amount of time spent in idleness by individual VMs, which in turn will improve the efficiency with which the system utilizes its resources. We assume that two tasks, u_k^j as well as $u_{k'}^j$, run within the identical VM V_{lm} , which means the duration required to complete u_k^j is shorter than the overall time it takes to initialize $u_{k'}^j$, which can be represented as $gu_{k,lm}^j < gu_{k',lm}^j$. Let's assume that somehow the VM V_{lm} has a startup duration of $iu_{q,lm}^{j'}$ for each workload $x_{j'}$, where $iu_{q,lm}^{j'}$ is smaller than $iu_{k',lm}^j$ for each of those workloads, then the $u_q^{j'}$ hold for the given (18).

$$\max \{gu_{k,lm}^j, iu_{q,lm}^{j'}\} + fu_{q,lm}^{j'} \leq iu_{k',lm}^j \quad (18)$$

Further, to minimize V_{lm} idle time, we divide $u_q^{j'}$ amongst u_k^j as well as $u_{k'}^j$. This presumption can be demonstrated by using Lemma 1. Consider u_k^j as well as $u_{k'}^j$ represent the active as well as idle time slots, respectively, for a specific VM V_{lm} . In (19) can be used to determine ω_1 , which represents the idle period.

$$\omega_1 = iu_{k',lm}^j - gu_{k,lm}^j \quad (19)$$

The new value for the shared idle time slot ω_2 between u_k^j and $u_{k'}^j$, is calculated as follows if the task $u_q^{j'}$ is distributed between them.

$$\omega_2 = iu_{k',lm}^j - gu_{k,lm}^j - fu_{q,lm}^{j'} \quad (20)$$

The preceding equation ensures that the corresponding VM V_{lm} will have less time spent waiting. According to Lemma 1, if many workload tasks are submitted to a single VM and run in parallel, the VMs idle time is cut

down, enabling better use of its resources. To satisfy application-deadlines as well as cut the amount of energy waste, the ERP-WSC approach scales up or down resources as needed to comply with Lemma 2. Consider a group of physical machines that are active represented as I_b and expressed as the given (21).

$$I_b = \{i_1, i_2, \dots, i_{|I_b|}\} \quad (21)$$

Where, the total computational capacity of a physical-machine is a constant, represented as (22).

$$G = \sum_{l=1}^{|I_b|} g_l \quad (22)$$

In (22), the g_l is used for defining the frequency of the physical-machine represented as $i_l \in I_b$. In (23) is used to determine the total amount of energy that is consumed by physical-machines defined as I_b .

$$E = \sum_{l=1}^{|I_b|} q_l, \quad (23)$$

If the following condition holds, then the preceding equation is minimal.

$$c_1 g_1^2 = c_2 g_2^2 = \dots = d_{|I_b|} g_{|I_b|}^2, \quad (24)$$

In (24), d_l is used for representing the physical-machine i_l which is static such that $1 \leq l \leq |I_b|$. Further, the d_l is evaluated by using the given (25).

$$d_l = \frac{(1-t_l) \cdot q_l^\uparrow}{(g_l^\uparrow)^3} \quad (25)$$

From (10), the consumption of energy q_l for the physical machine i_l which are active and can be calculated using the given (26).

$$q_l = t_l * q_l^\uparrow * z_l^u + \frac{(1-t_l) \cdot q_l^\uparrow}{(g_l^\uparrow)^3} * (g_l)^3 \quad (26)$$

Further, the given (26) can be simplified into the given (27).

$$q_l = t_l * q_l^\uparrow + d_k * (g_l^e)^3 \quad (27)$$

Here, the $d_k * g_l^2$ is stabilized over a wide variety of operational physical-machines in a heterogeneous cloud setting to guarantee minimum energy use. In this study, we present a dynamic re-planning method for workload execution that takes into account the criticality of meeting real-time deadlines. The ERP-WSC paradigm maximizes efficiency by balancing the use of both physical and virtual machines to ensure that application deadlines are met while reducing power usage. Meanwhile, with the current web service design for workload implementation, tasks are scheduled instantly to the corresponding virtual or physical machines. In contrast, the ERP-WSC only allocates tasks that can be run on the appropriate VMs. In addition, the unfinished tasks are saved in a queue, as well as the scheduling decisions for them are optimized, or re-planned, to make better use of time and resources.

The efficient-replanning algorithm for the web-service-composition has been given in Algorithm 1. The algorithm is dynamic and starts whenever a task arrives. The tasks are then kept on hold inside the *Queue*. Further, all the scheduling decisions are kept on hold and the current status of the virtual machines is attained. After this, the ru_k^j for each task is evaluated utilizing (20). After the evaluation, the tasks are further added to the *Queue*. Further, the tasks are allocated with optimized resources and the energy is reduced from Step 8 to Step 20. Moreover, the tasks having higher priority are first sent into the *Queue*. For reducing energy, the algorithm ensures that appropriate resources are given to the required tasks. If a greater number of virtual machines are required for the execution of the tasks, then the resources are scaled up. If the resources are not utilized by the tasks, then the resources are scaled down for reducing energy consumption. The process of scaling down and up the resources according to the requirement of the workload tasks helps us to attain good performance. The performance of the proposed ERP algorithm has been evaluated in the next section.

Algorithm 1. ERP algorithm for WSC

```

Step 1. Start
Step 2. Queue ← ∅;
Step 3. ∀ workload  $x_j$  arrival do
Step 4. Stop the non-executed scheduling decisions;
Step 5. Collect the status and inform available virtual machines;
Step 6. Compute  $ru_k^j$  for every task of workload using Eq. (20);
Step 7. Add the complete task of the workload  $x_j$  into Queue;
Step 8. While Queue composed of unscheduled tasks do
Step 9.  $\mathcal{R} \leftarrow$  obtain tasks from Queue that are ready;
Step 10. Arrange  $\mathcal{R}$  in ascending order (i.e., increasing) concerning  $ru_k^j$ ;
Step 11.  $\forall u_k^j \in \mathcal{R}$  do
Step 12. ChoseVM ← ∅; minW ← ∞;
Step 13. CVMs ← all virtual machines satisfying  $g_{l,m} > cpu_k^j$ ;
Step 14.  $\forall v_{l,m} \in CVMs$  do
Step 15. Compute resource wastage  $\mathcal{W}_{k,l,m}^j$  of  $u_k^j$  and completion time  $gu_{k,l,m}^j$  on  $v_{l,m}$ ;
Step 16. If  $gu_{k,l,m}^j \leq ru_k^j < minW$  then
Step 17. ChoseVM ←  $v_{l,m}$ ; minW ←  $\mathcal{W}_{k,l,m}^j$ ;
Step 18. If ChoseVM == ∅ then
Step 19. ChosenVM ← Resl;
Step 20. Schedule task  $u_k^j$  to ChoseVM;
Step 21. Stop.
    
```

4. RESULTS AND DISCUSSION

The results have been discussed in this section. For evaluating the proposed ERP, Montage workflow has been used and then compared with the current QL-HEFT method [24]. More description of the Montage workflow can be attained from [25]. The results have been evaluated in terms of time required for the execution of each Montage task, total power sum consumed during the execution of the Montage tasks, total power average consumed during the execution of the Montage tasks, and total energy consumed for the execution of the Montage tasks. All these have been explained in below subsections. For the execution of the 100 Montage tasks, 40 and 60 physical machines have been considered. The virtual machines have been varied from 10, 15, 20, and 25 during the execution of the 100 Montage tasks.

4.1. Overall time required for the execution of the Montage tasks

The overall time required for the execution of the Montage tasks has been discussed in this section. In Figures 1 and 2, it can be seen that the heterogeneous earliest finish time (QL-HEFT) method takes more time for the execution of the 100 Montage tasks using 40 as well as 60 physical machines, respectively. Moreover, when the virtual machines are increased, the QL-HEFT method still fails to reduce the execution time for the execution of 100 Montage tasks. Further, the proposed ERP method executes the 100 Montage tasks in less time and when the virtual machines are increased, the ERP method decreases the execution time. From both results, it can be said that the proposed ERP is more efficient for the execution of scientific workloads.

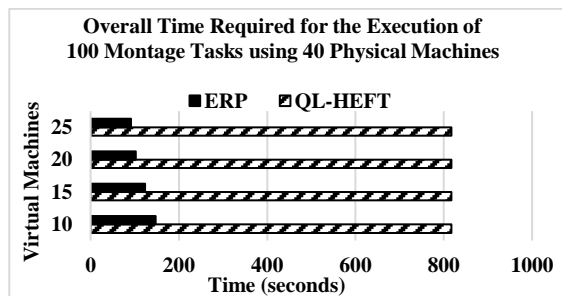


Figure 1. Overall time required for the execution of 100 Montage tasks using 40 physical machines

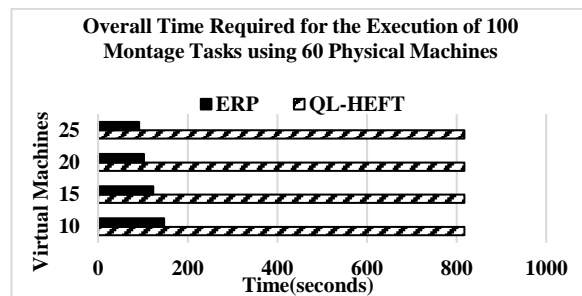


Figure 2. Overall time required for the execution of 100 Montage tasks using 60 physical machines

4.2. Total power sum consumed for the execution of the Montage tasks

The total power sum consumed for the execution of the Montage tasks has been discussed in this section. In Figures 3 and 4, it can be seen that the QL-HEFT method consumes more power sum for the

execution of the 100 Montage tasks using 40 as well as 60 physical machines, respectively. Moreover, when the virtual machines are increased, the QL-HEFT method still fails to reduce the power sum for the execution of 100 Montage tasks. Further, the proposed ERP method executes the 100 Montage tasks in less power sum and when the virtual machines are increased, the ERP method decreases more power sum. From both results, it can be said that the proposed ERP is more efficient and reliable for the execution of scientific workloads.

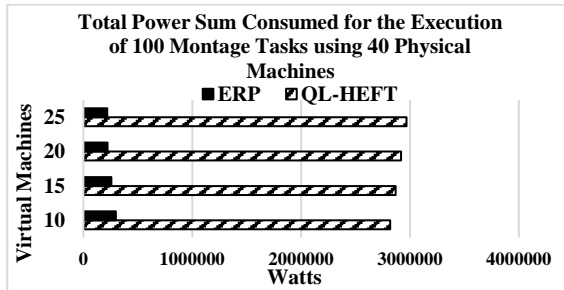


Figure 3. Total power sum consumed for the execution of 100 Montage tasks using 40 physical machines

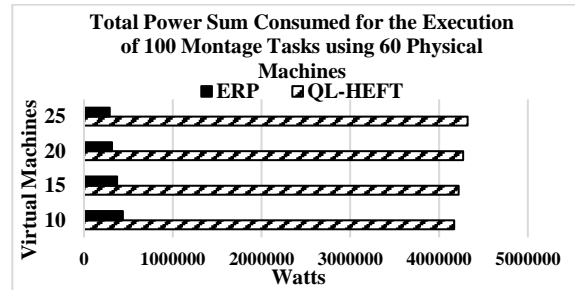


Figure 4. Total power sum consumed for the execution of 100 Montage tasks using 60 physical machines

4.3. Total power average consumed for the execution of the Montage tasks

The total power average consumed for the execution of the Montage tasks has been discussed in this section. In Figures 5 and 6, it can be seen that the QL-HEFT method consumes more power on average for the execution of the 100 Montage tasks using 40 as well as 60 physical machines, respectively. Moreover, when the virtual machines are increased, the QL-HEFT method still fails to reduce the power average for the execution of 100 Montage tasks. Further, the proposed ERP method executes the 100 Montage tasks with less power average and when the virtual machines are increased, the ERP method decreases more power average. From both results, it can be said that the proposed ERP is more efficient and reliable for the execution of scientific workloads.

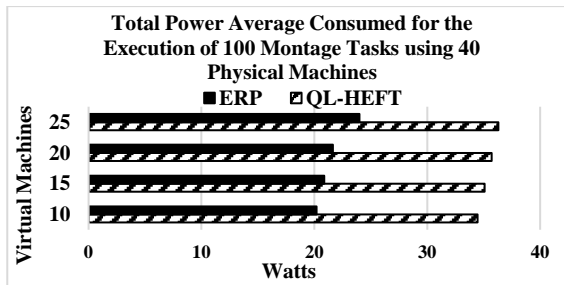


Figure 5. Total power average consumed for the execution of 100 montage tasks using 40 physical machines

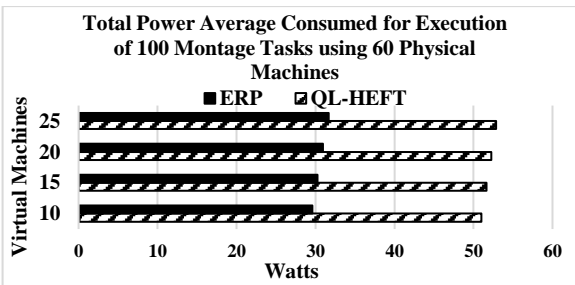


Figure 6. Total power average consumed for execution of 100 Montage tasks using 60 physical machines

4.4. Total energy consumption consumed for the execution of the Montage tasks

The total energy consumption consumed for the execution of the Montage tasks has been discussed in this section. In Figures 7 and 8, it can be seen that the QL-HEFT method consumes more energy for the execution of the 100 Montage tasks using 40 as well as 60 physical machines, respectively. Moreover, when the virtual machines are increased, the QL-HEFT method still fails to reduce the energy for the execution of 100 Montage tasks. Further, the proposed ERP method executes the 100 Montage tasks with less energy and when the virtual machines are increased, the ERP method increases the energy slightly due to the increase in the virtual machines. From both results, it can be said that the proposed ERP is more efficient and reliable for the execution of scientific workloads.

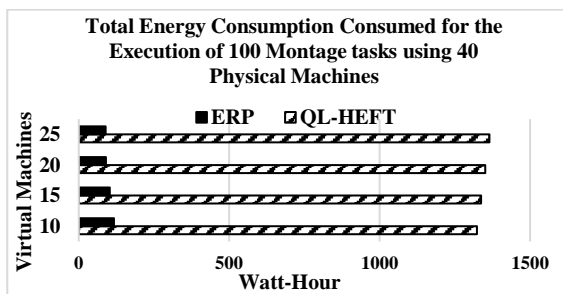


Figure 7. Total energy consumption consumed for the execution of 100 Montage tasks using 40 physical machines

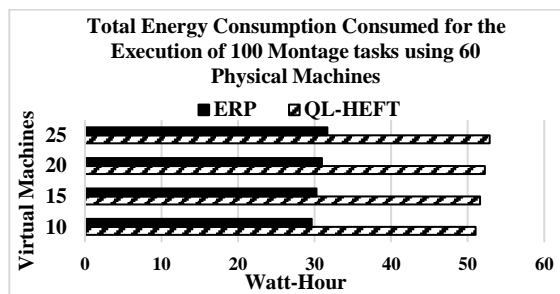


Figure 8. Total energy consumption consumed for the execution of 100 Montage tasks using 60 physical machines

5. CONCLUSION

The efficient replanning web service composition algorithm for workload scheduling under a cloud environment proposed in the research presents a promising approach to optimize the execution of complex workflows in cloud environments. The algorithm provides an efficient method to tackle the challenge of workload scheduling in cloud environments, which is critical for achieving efficient and reliable cloud services. The proposed algorithm considers various factors to generate an optimal schedule for the given workflow. The simulation results demonstrate that the proposed ERP algorithm outperforms the existing QL-HEFT algorithm in terms of total execution time, total energy consumption, total power average, and total power sum. The approach's effectiveness is further validated through a case study that involves the execution of a complex workflow. Overall, the proposed algorithm can significantly improve the efficiency and reliability of cloud services, making it a valuable contribution to the field of cloud computing. For future work, this work can be extended by executing other scientific workloads.





REFERENCES

- [1] R. Tasnim, A. A. Mim, S. H. Mim, and P. D. M. I. Jabillah, "A comparative study on three selective cloud providers," *SSRN Electronic Journal*, 2022, doi: 10.2139/ssrn.4205141.
- [2] L. Yang, C. Zhong, Q. Yang, W. Zou, and A. Fathalla, "Task offloading for directed acyclic graph applications based on edge computing in Industrial Internet," *Information Sciences*, vol. 540, pp. 51–68, Nov. 2020, doi: 10.1016/j.ins.2020.06.001.
- [3] X. Feng, C. Shushan, H. Xingxing, H. Shujuan, and Z. Wenjuan, "A new direct acyclic graph task scheduling method for heterogeneous Multi-Core processors," *Computers and Electrical Engineering*, vol. 104, Dec. 2022, doi: 10.1016/j.compeleceng.2022.108464.
- [4] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 2, pp. 149–158, Feb. 2020, doi: 10.1016/j.jksuci.2018.01.003.
- [5] P. Loncar and P. Loncar, "Scalable management of heterogeneous cloud resources based on evolution strategies algorithm," *IEEE Access*, vol. 10, pp. 68778–68791, 2022, doi: 10.1109/ACCESS.2022.3185987.
- [6] A. A. Khan, M. Zakarya, I. U. Rahman, R. Khan, and R. Buyya, "HeporCloud: An energy and performance efficient resource orchestrator for hybrid heterogeneous cloud computing environments," *Journal of Network and Computer Applications*, vol. 173, Jan. 2021, doi: 10.1016/j.jnca.2020.102869.
- [7] H. Lee, S. Cho, Y. Jang, J. Lee, and H. Woo, "A global DAG task scheduler using deep reinforcement learning and graph convolution network," *IEEE Access*, vol. 9, pp. 158548–158561, 2021, doi: 10.1109/ACCESS.2021.3130407.
- [8] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "Multiobjective task scheduling in cloud environment using decision tree algorithm," *IEEE Access*, vol. 10, pp. 36140–36151, 2022, doi: 10.1109/ACCESS.2022.3163273.
- [9] S. Kapoor and S. N. Panda, "Scheduling of parallel tasks in cloud environment using DAG MODEL," in *Advances in Intelligent Systems and Computing*, vol. 1172, 2021, pp. 267–276, doi: 10.1007/978-981-15-5566-4_23.
- [10] S. P. M. Ziyath and S. Senthilkumar, "MHO: meta heuristic optimization applied task scheduling with load balancing technique for cloud infrastructure services," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 6, pp. 6629–6638, Jun. 2021, doi: 10.1007/s12652-020-02282-7.
- [11] P. Singh, V. Prakash, G. Bathla, and R. K. Singh, "QoS aware task consolidation approach for maintaining SLA violations in cloud computing," *Computers and Electrical Engineering*, vol. 99, p. 107789, Apr. 2022, doi: 10.1016/j.compeleceng.2022.107789.
- [12] C. Zhang, Y. Wang, Y. Lv, H. Wu, and H. Guo, "An energy and sla-aware resource management strategy in cloud data centers," *Scientific Programming*, vol. 2019, pp. 1–16, Nov. 2019, doi: 10.1155/2019/3204346.
- [13] C. Zhang, Y. Wang, H. Wu, and H. Guo, "An energy-aware host resource management framework for two-tier virtualized cloud data centers," *IEEE Access*, vol. 9, pp. 3526–3544, 2021, doi: 10.1109/ACCESS.2020.3047803.
- [14] 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.
- [15] A. Vafamehr and M. E. Khodayar, "Energy-aware cloud computing," *Electricity Journal*, vol. 31, no. 2, pp. 40–49, Mar. 2018, doi: 10.1016/j.tej.2018.01.009.
- [16] G. Xie, L. Liu, L. Yang, and R. Li, "Scheduling trade-off of dynamic multiple parallel workflows on heterogeneous distributed computing systems," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 2, Jan. 2017, doi: 10.1002/cpe.3782.
- [17] K. Khorramnejad, L. Ferdouse, L. Guan, and A. Anpalagan, "Performance of integrated workload scheduling and pre-fetching in multimedia mobile cloud computing," *Journal of Cloud Computing*, vol. 7, no. 1, Dec. 2018, doi: 10.1186/s13677-018-0115-6.





- [18] 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.
- [19] 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.
- [20] 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.
- [21] M. U. Sana and Z. Li, "Efficiency aware scheduling techniques in cloud computing: A descriptive literature review," *PeerJ Computer Science*, vol. 7, pp. 1–37, May 2021, doi: 10.7717/PEERJ-CS.509.
- [22] 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.
- [23] P. Neelima and A. R. M. Reddy, "An efficient load balancing system using adaptive dragonfly algorithm in cloud computing," *Cluster Computing*, vol. 23, no. 4, pp. 2891–2899, Dec. 2020, doi: 10.1007/s10586-020-03054-w.
- [24] 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.
- [25] IRSA, "Parallelization and performance. Montage user documentation," NASA Space Act Award Winner 2006, 2006, accessed Dec. 27, 2022. [Online]. Available: <http://montage.ipac.caltech.edu/docs/grid.html>.

BIOGRAPHIES OF AUTHORS



Kavita D. Hanabaratti     was born in Karnataka, India, in 1979. She received a B.E. degree in Computer Science and Engineering from the University of KUD, in the year 2001 and an M. Tech in Computer Science and Engineering from the Gogte Institute of Technology (GIT) Belagavi India, in 2011. From 2002 to 2004 she served as a lecturer at SDM Engineering College Dharwad. She is currently working as an Assistant Professor in the Department of Computer Science and Engineering at Gogte Institute of Technology Belagavi since 2007. Her current research interests include autonomic computing, machine learning, and artificial intelligence. She is a Life Member of the Indian Society for Technical Education (ISTE). She can be contacted at email: kdhanabaratti@git.edu.



Dr. Rudragoud Patil     currently working as an Associate Professor, at Department 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.