Received: June 13, 2023. Revised: September 16, 2023. 118

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Quality of Service Aware Task Replanning Algorithm for Web-Service Provisioning in Cloud Environment

Kavita D. Hanabaratti[1]* **Rudragoud Patil[1]**

[1]*Department of Computer Science and Engineering, Karnataka Law Society's Gogte Institute of Technology, Visvesvaraya Technological University, Belagavi – 590018, India*
* Corresponding author's Email: kdhanabaratti@git.edu

Abstract: In web service composition, the combination of multiple individual services to create complex workflows introduces additional challenges, and task scheduling is a crucial aspect that significantly impacts the overall system performance. The main challenge in web service composition task scheduling arises from the varying nature of services, which may have diverse processing requirements, data dependencies, and execution times. Hence, to address these challenges, the existing work have proposed many methods. But, as cloud environments often host these web services, the dynamic nature of cloud resources and the unpredictability of workloads add another layer of complexity to the task scheduling process. Due to this the existing works have attempted very less work on providing an efficient replanning task based on real-time changes in resource availability. Hence, this paper proposes a novel approach to cloud workload task scheduling in web service composition that incorporates efficient replanning and time complexity analysis called as Web service composition-efficient re-planning (WSC-ERP). The proposed approach takes into account various factors, such as task dependencies, and resource availability, to generate an optimized schedule that minimizes execution time and maximizes resource utilization. The WSC-ERP has been evaluated using the Montage scientific workload. The results show that the WSC-ERP provides better performance in terms of executional time, power sum, power average, energy consumption and reliability when compared with the Energy-Minimized Scheduling (EMS), and Evolutionary Computing based Web Service Composition (EC-WSC). The results show that the WSC-ERP has showed an improvement of 80.27% and 78.45% for average execution time, 89.98% and 87.49% for average power sum, 63.44% and 41.93% for average power average, 92.58% and 87.87% for average energy consumption, 4.97% and 4.07% for average reliability when compared with the existing EMS and EC-WSC models respectively.

Keywords: Web-service composition, Task dependency, Resource availability, Resource utilization, montage.

1. Introduction

Workload scheduling in the cloud involves the management and allocation of resources to different workloads or tasks that are running on a cloud computing platform. This process is important for optimizing resource utilization and ensuring that all tasks are completed efficiently and within their desired timelines [1]. There are different approaches to workload scheduling in the cloud, including rule-based scheduling [2], resource-based scheduling [3], and priority-based scheduling [4]. Cloud workload scheduling also involves the use of automation and orchestration tools to automate the scheduling and management of resources. These tools can help to optimize resource usage, reduce costs, and improve the overall efficiency of cloud workloads. In addition, workload scheduling in the cloud requires monitoring and analysis of workload performance and resource utilization [5]. Effective workload scheduling in the cloud is essential for achieving optimal resource utilization, improving workload performance, and ensuring that tasks are completed on time and within budget.

Workload scheduling is a critical component of web server composition in the cloud. It involves managing the allocation of resources to multiple web

servers to ensure optimal performance, availability, and reliability of web services [6]. However, workload scheduling for web server composition can be challenging due to the variability of web traffic patterns, the need to balance the load across multiple servers, and the choice of architecture. The workload scheduling process for web server composition can be complex due to a number of issues and challenges. One of the main challenges in workload scheduling for web server composition is the variability in web traffic patterns [7]. Web traffic can be highly unpredictable and can fluctuate significantly over short periods of time. This can make it difficult to allocate resources effectively and ensure that all web servers are able to handle the workload. Another challenge is the need to balance the load across multiple web servers [8]. In order to achieve optimal performance and avoid overloading any one server, it is important to distribute the workload evenly across all servers. This requires sophisticated load balancing algorithms and techniques that can handle varying traffic patterns and dynamically adjust resource allocation [9]. The choice of web server composition architecture also presents challenges for workload scheduling. Different architectures, such as monolithic or microservices, can have different requirements in terms of resource allocation and workload scheduling [10]. It is important to select the right architecture for the specific workload and to ensure that the workload scheduling process is optimized accordingly [11], [12]. In summary, workload scheduling for web server composition in the cloud is a complex process that requires careful consideration of a range of factors, including traffic variability, energy consumption, deadline execution and load balancing. Effective workload scheduling can help to optimize resource utilization, improve performance, and ensure the availability and reliability of web services.

As most of the exiting works have addressed to solve the issue of traffic variability, energy consumption, deadline execution and load balancing, yet, these existing works have not presented an efficient replanning method to plan the sequence in which the workload will be executed. Hence, the contribution of the proposed work are as follows

- Present a model for workload task-scheduling in cloud environment which will execute the tasks in the given deadline and reduces the resource utilization and energy consumption.
- Present an Efficient Re-Planning (ERP) method to plan the sequence in which the workload will be executed.
- Evaluate the proposed model using QoS metric such as time, energy consumption and reliability and compare it with other existing models.

The paper has been organized in the following sections. In the section II, literature survey has been given where different research work for the efficient execution of the workload have been studied. In the section 3, the proposed Web-service composition (WSC) and efficient re-planning (ERP) method has been presented. In the section 4, the results for the proposed method have been evaluated in terms of execution time, power sum, power average, energy consumption, and reliability and compared with EMS and EC-WSC. Finally, in the section 5, the conclusion of the work along with the future work has been given.

2. Literature survey

This section provides a survey on the recent work presented by various researchers for the efficient execution of the workload. In [13], this research offers a complete hybrid workload scheduling method, namely HPCP-PSO, to tackle cloud workload scheduling by formulating it in the context of a constrained optimizing issue which maximizes workflow execution costs under a workflow time constraint. By combining the algorithm known as IaaS cloud-partial critical-paths (IC-PCP) with the meta-heuristic particle-swarm-optimization (PSO), HPCP-PSO achieves better results than prior efforts by utilizing a hybrid approach. In-depth investigations using four actual scientific workloads demonstrate that the suggested method reduces the average cost of executing the workload by 87.71%, 70.53% and 35.83% in comparison to HGSA, PSO, and IC-PCP respectively. In [14], to reduce the time and energy required to complete a workload application using microservices while still meeting strict deadlines and quality standards, this work presents the GSMS heuristic algorithm. To guarantee sub-reliability, GSMS uses a greedy fault-tolerant scheduling method, in which copies of every task are scheduled using the least expensive and most efficient resources available. In addition, GSMS incorporates a resource modification mechanism that will further enhance the use of resources. Comprehensive research using a number of actual workload applications, compared to pre-existing algorithms, show GSMS's efficiency and effectiveness in reducing processing cost while simultaneously satisfying reliability and deadline constraints.

In [15], in this study, they suggest several variants of the HEFT algorithm that have been modified to yield better outcomes. They use multiple methods to generate ranks in the initial phase (rank-generation), and then modify how available timeslots are chosen in the subsequent phase (task-scheduling). Based on their research, they conclude that modified HEFT algorithms outperform the original HEFT algorithm when it comes to shortening the duration of workflow schedules. In [16], the suggested methodology uses multistage-forward search (MSF) to improve the process of web service identification as well as composing by reducing the total number of incorporated Web-services. The recommended approach also employs the spider-monkey optimization (SMO) algorithm, that improves the quality of the services issue by providing to each of the symmetrical and asymmetrical aspects of service composition. The suggested model's superiority, durability, and feasibility has been demonstrated through a comparison of the experiment's outcomes against the underlying findings of the smart-multistage-forward search (SMFS) approach. Based on findings from experiments, the suggested SMO method reduces service composition creation time by 40 percent comparison to the SMFS method.

Cloud computing task scheduling has been hampered by issues including excessive wait times, excessive resource usage, and overloaded virtual machines. To combat these issues, an enhanced scheduler effectiveness algorithm called as wild horse optimization (IWHO) is developed in [17]. A system for the planning and allocation of cloud computing tasks has been constructed. The major considerations in this model are virtual machines, time and cost. The effectiveness of the IWHOLF-TSC technique has been verified, and the outcomes have been assessed in a number of ways. The IWHOLF-TSC method was found to be superior to others in a number of simulated scenarios. In [18], they presented a model called a RACES for the heterogenous computing environment. This work mainly focussed to reduce the cost and time for the execution of the workloads. In this model they have used Weibull-distribution method as optimization strategy. In this model, for evaluating their model, they have used complex workload type. Further, in [19], they presented a model called as ARPS for the cloud environment. This work mainly focussed to reduce the cost, time and energy consumption during the execution of the workloads. The model has used Spider Monkey Optimization method as their optimization strategy. The model used simple workload for evaluating their model. In [20], they have presented a MOWOS model which addresses the time, budget and deadline challenges during the execution of the workloads. Also, this work mainly focussed on the cloud environment. The model has used Heuristic method for optimizing these issues. In this work they have used complex workload type for evaluating their model. In [21], they have presented a method called as EMS which focusses on the heterogenous computing environment. This work main focus was to reduce the energy consumption during the execution of the workloads. They have used a heuristic approach for optimizing the issue of energy consumption. In [22], they have presented a model called as DB-ACO which addresses the cost, budget and deadline during the execution of the workload. They have used Ant colony optimization method to optimize all these challenges.

In summary, [18], [21], and [22] have used heterogenous computing environment, whereas all the existing work mainly focussed on the cloud environment. Also, all the research work focussed to analyse their models using complex workload types. Different research works have addressed different challenges such as cost, time, deadline, energy consumption. Further, all the methods have used an optimization strategy for addressing the challenges faced during the execution of the workload. Only the [20] research work has presented a replanning method whereas all the existing works have not presented any replanning method. Also, only [21] research work focussed to provide reliability. Hence, in the proposed WSC-ERP model, we focus on the heterogenous computing environment. This work focusses to reduce the energy consumption, resource utilization, and execution time. In this work we have used a heuristic optimization strategy for optimizing all these issues. Finally, the WSC-ERP model provides a Re-planning algorithm for scheduling the workload tasks which will in turn provide better reliability.

3. Model

The notations used in the model have been given in the Table 1.

3.1 Workload model

The workload which needs to be scheduled in the cloud environment can be represented using the given equation

$$X = \{x_1, x_2, x_3, \dots, x_n\}, \quad (1)$$

where, $x_j (j = 1,2, \dots n)$ is used for defining the j^{th}

Table 1. Notation table

Equation	Notation
X	Workload
x_j	j^{th} workload where $(j = 1,2, \dots n)$
b_j	Time at which the workload will arrive
e_j	Deadline time in which the workload has to be executed
H_j	Structure of the workload
U_j	Various group of tasks present inside the x_j and F_j
f_{qk}^j	Edge of the DAG
F_j	$f_{qk}^j \in F_j$, starting state of task u_k^j which is reliant on the output from the task u_q^j
u_k^j	Forthcoming-task
u_q^j	Preceding-task
o	Physical-machine size
xt	Start-time to execute the workload
yt	End-time to execute the workload
t_l	Static energy-consumption
$q_l^{\uparrow}$	Highest level of energy
z_l^u	Current state of the Physical-Machine
$g_l^{\uparrow}$	Highest frequency-level of CPU
n	Workload-size of X
$\lvert U_j \rvert$	Various group of tasks present inside the x_j
cpu_k^j	CPU frequency of the forthcoming-task u_k^j
$\mathcal{T}_k^j$	Time necessary for the execution of the workload tasks
$\mathcal{A}_l$	Time that a physical machine has been in the active state
ru_k^j	Recent starting time for the execution of the tasks of the workload
e_j	Deadline-time for the execution of the task of the workload
du_k^j	Deadline for the forth-coming task u_k^j
$succ(u_k^j)$	Overall task of the workload which has the forth-coming task u_k^j
AH	Active PM with unutilized CPU frequency which is greater than $g_l^{\uparrow}$
vm_l	Virtual Machine
h_k	Physical Machine
c_k	Constant for PM h_k
$\frac{p_k^{max}}{g_k^{max}}$	Energy-Frequency Ratio
$\lvert X \rvert$	Number of tasks present in workload X
$\max_{x_j \in X}\{\lvert U_j \rvert\}$	Maximum task size
$\max_{x_j \in X}\{\lvert F_j \rvert\}$	Maximum edges present in workload X
H_V	Total number of virtual machines
H_I	Total number of physical machines

workload. The x_j is defined as the following equation

$$x_j = \{b_j, e_j, H_j\}, \tag{2}$$

where, b_j is used for defining the time at which the workload will arrive, e_j is used for defining the deadline time in which the workload has to be executed, H_j is used for defining the structure of the workload. Workloads might have various structures; hence, they can be represented using a directed-acyclic-graph (DAG) by using the given equation

$$H_j = (U_j, F_j) \tag{3}$$

where, U_j is used for defining the various group of tasks present inside the x_j and F_j is used for representing the data-dependencies present amongst the x_j. The U_j and F_j have been defined in the Eq. (4) and Eq. (5) respectively.

$$U_j = \left\{u_1^j, u_2^j, \dots, u_{|U_j|}^j\right\} \tag{4}$$

$$F_j \subseteq U_j \times U_j \tag{5}$$

The edge of the DAG represented f_{qk}^j belongs to the F_j, i.e., $f_{qk}^j \in F_j$. Further, the $f_{qk}^j \in F_j$ is used for describing the starting state of task u_k^j which is reliant on the output from the task u_q^j which is currently being executed on the cloud. Thus, the forthcoming-task is represented as u_k^j and the preceding-task is represented as u_q^j.

3.2 Model for workload task-scheduling in web-service-composition (WSC)

Workload task-scheduling in WSC refers to the process of assigning tasks or sub-tasks of a composite web service to appropriate service providers or resources in order to optimize performance, efficiency, and cost. When a composite service is executed, it is often necessary to break down the service into smaller tasks or sub-tasks that can be performed by different service providers or resources. These tasks can be interdependent, and may have different resource requirements, priorities, and deadlines. Workload task scheduling involves determining the optimal order and assignment of these tasks to the available service providers or resources based on various factors such as availability, cost, response time, and quality of service (QoS) requirements. The cloud enables WSC

to be performed on a large scale, by leveraging the vast resources and capabilities of cloud service providers (CSPs). To schedule the tasks of the workload in order to reduce the execution time and energy consumption, the proposed model satisfies the constraints which have been presented in [23], [24]. After satisfying all the constraints, the following equation is obtained.

$$\alpha = \text{Min} \sum_{l=1}^{o} \int_{xt}^{yt} \left(t_l * q_l^{\uparrow} * z_l^u + \frac{(1-t_l)*q_l^{\uparrow}}{(g_l^{\uparrow})^3} * (g_l^e)^3 \right) dt. \quad (6)$$

where, o is used for representing the physical-machine size, xt and yt are used for representing start and end-time to execute the workload, t_l is used for representing the static energy-consumption, $q_l^{\uparrow}$ is used for representing the highest level of energy, z_l^u is used for representing the current state of the physical-machine (PM), and $g_l^{\uparrow}$ is used for representing the highest frequency-level that a CPU can operate. Further, for allocating the efficient resources for the execution of the workload tasks, the given equation is used.

$$\beta = \text{Max} \frac{\left(\sum_{j=1}^{n} \sum_{k=1}^{|U_j|} cpu_k^j * \mathcal{T}_k^j \right)}{\left(\sum_{l=1}^{o} g_l^{\uparrow} * \mathcal{A}_l \right)} \quad (7)$$

where, n is used for defining workload-size of X, $|U_j|$ is used for defining the various group of tasks present inside the x_j, cpu_k^j is used for representing the CPU frequency of the forthcoming-task u_k^j, $\mathcal{T}_k^j$ is used for representing the time necessary for the execution of the workload tasks, o is used for representing the physical-machine size, $g_l^{\uparrow}$ is used for representing the highest frequency-level that a CPU can operate and $\mathcal{A}_l$ is used for representing the time that a physical machine has been in the active state.

3.3 Efficient re-planning (ERP) method for WSC in cloud environment

In this section, the ERP method for the workload scheduling for the WSC in the cloud environment has been presented. In this proposed method, the workload tasks are scheduled utilizing the Min-Max model presented in the Eqs. (6) and (7). Nevertheless, to find the best resources for the workload task execution, the Eq. (6) and Eq. (7) has an effect on the various parameters like resource availability, deadline-prerequisite (reliability), and task dependency. Hence, to schedule the tasks of the workload by utilizing the Mix-Max model can be said to be a NP-hard problem. In this proposed model, a heuristic approach for solving the trade-off issue and for attaining the best solution having time and reliability requirements in the context of executing a dynamic workload has been presented. In the traditional models which have been proposed, it is seen that the workload tasks are given as an input to the resource provisioner, which then schedules only some of those tasks to individual virtual machines and leaves the rest of the tasks waiting. The previous models used to schedule the tasks depending on the task-dependency. When a new workload arrives, the traditional model failed to provide resource to the workload tasks as the other tasks are waiting. Hence, due to this the reliability of the task is affected and there is wastage of resources. Therefore, there is a requirement for an ERP method which can be used for optimizing the execution of the workload when a new workload arrives. Planning the sequence in which workloads will be carried out therefore becomes essential in this proposed work prior to scheduling their execution. For ERP method to be effective, first, we consider the most recent starting time ru_k^j of each workload. Further, the tasks of the workload are considered which are later on ranked for the workload task-execution. The ru_k^j of every task u_k^j is defined as the recent-time, before the initialization of the u_k^j. If the execution of the task fails, then, completion-time gu_k^j of x_j can have a small delay, misses the deadline for the execution of the task, and has a big impact on the reliability for the workload task-scheduling. Further, the ru_k^j can be evaluated by the given equation

$$r\left(u_k^j\right) = \begin{cases} e_j - du_k^j, & \text{if } succ\left(u_k^j\right) = \emptyset \\ \min\limits_{u_s^r \in \left(u_k^j\right)} \left\{ \left(ru_s^j\right) - uu_{ks}^j - du_k^j \right\}, & \text{otherwise.} \end{cases} \quad (8)$$

where, ru_k^j is used for defining the recent starting time for the execution of the tasks of the workload, e_j is used for representing the deadline-time for the execution of the task of the workload, du_k^j is used for defining the deadline for the forth-coming task u_k^j, and $succ\left(u_k^j\right)$ is used for defining the overall task of the workload which has the forth-coming task u_k^j.

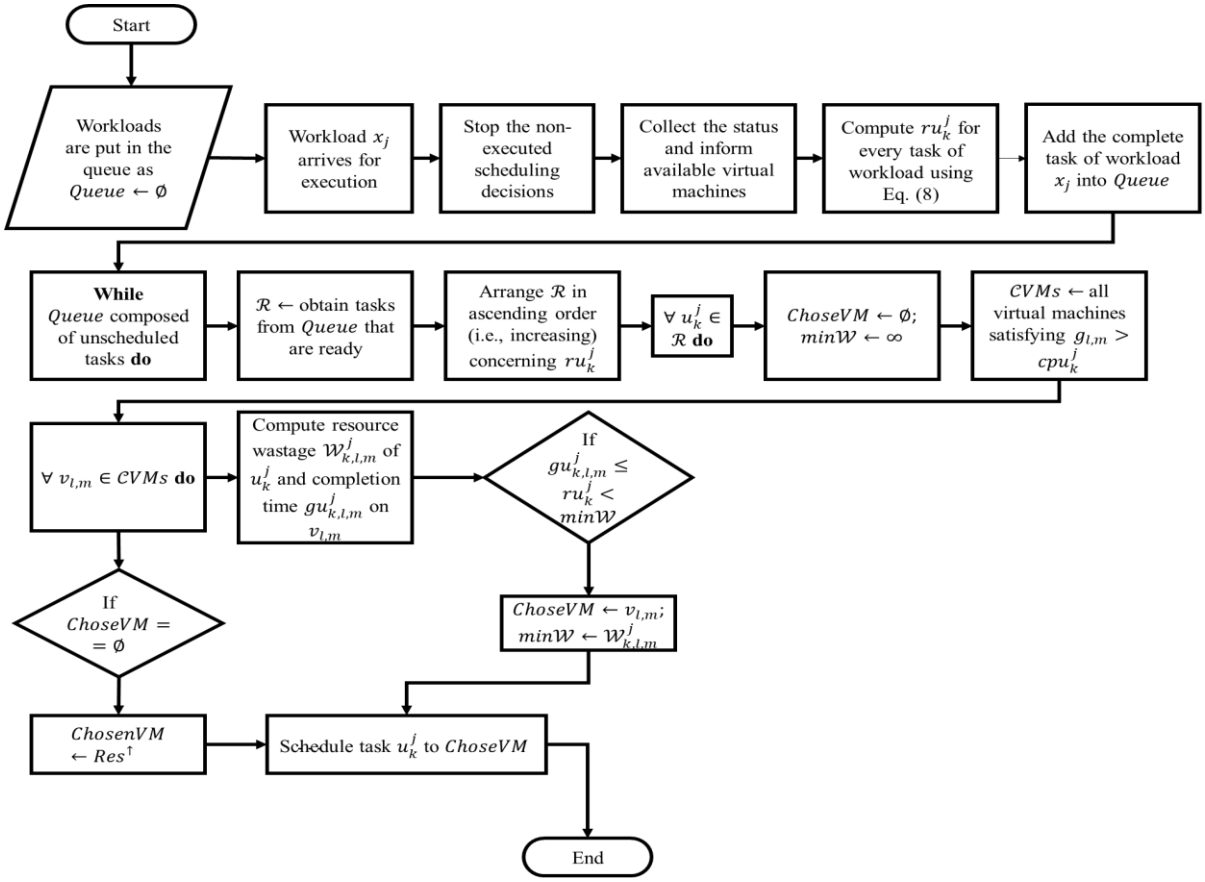

Figure. 1 Flowchart of the ERP method

Algorithm 1: Resource scaling ($Res^{\uparrow}$)	
Step 1.	Select a VM vm_l which has CPU frequency greater than the cpu_k^j;
Step 2.	AH ←active PM with unutilized CPU frequency which is greater than $g_l^{\uparrow}$
Step 3	destPM←NULL; minValue←$+\infty$;
Step 4.	Foreach $h_k \in$ AH do
Step 5.	If $c_k \cdot g_k^2 <$ minValue then
Step 6.	destPM←h_k; minValue← $c_k \cdot g_k^2$
Step 7.	If desPM! = NULL then
Step 8.	Initiate the VM $vm_{k,l}$ on destPM; Return $vm_{k,l}$;
Step 9.	else
Step 10.	OH←off PMs in the data-center;
Step 11.	Foreach $h_k \in$ OH do
Step 12.	If $g_k^{max} > cpu_j^i$ & $\frac{p_k^{max}}{g_k^{max}} <$ minValue then
Step 13.	desPM←h_k; minValue←$\frac{p_k^{max}}{g_k^{max}}$
Step 14.	Turn on host desPM, then initiate VM $vm_{k,l}$ on it;
Step 15.	Return $vm_{k,l}$;

In Algorithm 1, first, a VM set represented as vm_l is selected which meets u_k^j CPU frequency prerequisite (Step 1). For reducing the overall energy consumption, the PM first ensures that the vm_l has the least $c_k \cdot g_k^2$ and then the vm_l is selected for deployment of a new VM (Step 2 to Step 6). If the (Step 2 to Step 6) is not feasible, then an off PM having least energy-frequency ratio is selected which can contain the vm_l is switched on for the new VM (Step 9 to Step 15).

3.4 Time complexity analysis

The time complexity of the ERP-WSC algorithm for the execution of workload X is mathematically presented in the below equation

$$\gamma = O\left(|X| \cdot \max_{x_j \in X}\{|U_j|\} \cdot \left(\max_{x_j \in X}\{|F_j|\} + H_V + H_I\right)\right) \quad (9)$$

where $|X|$ defines the number of tasks present in workload X, $\max_{x_j \in X}\{|U_j|\}$ defines the maximum task size, $\max_{x_j \in X}\{|F_j|\}$ defines maximum edges present in workload X, H_V defines the total number of virtual

machines and H_I defines the total number of physical machines.

The time complexity of Algorithm 1 defined in Eq. (9) is proved as follows. In Algorithm 1, to compute the recent start time it takes $O(|U_j||F_j|)$. Further, for validating if the virtual machine can execute a ready task it takes $O(H_V)$. Then finding an ideal active physical machine takes $O(H_a)$, where H_a defines the number of active physical machines. Similarly, the number of the switched-off physical machine is defined as H_{off} and time taken to the switch-off physical machine is defined as $O(H_{off})$. Therefore, scaling up the resource takes $O(H_a + H_{off}) = O(H_I)$. Therefore, the total complexity of the proposed model is $O(H_V + H_I)$ for scheduling the task that is ready for the virtual machine. Alongside, it takes $O\left(|U_j||F_j| + |U_j|(H_V + H_I)\right)$ for scheduling the entire task of workload x_j. Thus, the time complexity is computed using the following equation

$$O\left(|X| \cdot \max_{x_j \in X}\{|U_j|\} \cdot \left(\max_{x_j \in X}\{|F_j|\} + H_V + H_I\right)\right) = O\left(|X| \cdot \max_{x_j \in X}\{|U_j|\} \cdot \left(\max_{x_j \in X}\{|F_j|\} + H_V + H_I\right)\right) \quad (10)$$

On the other side, the EMS [21] induces a time complexity of $O(M \times N^2 \times \log(w_{max}))$, where M represents processor size, N represents task size, and w_{max} defines the maximum value of computation cost matrix W. Similarly, the RACES [18] induce time complexity of $O(\zeta n^2)$, where n^2 defines time taken for sorting and ζ defines time taken for obtaining the optimal solution. The time complexity of MOWOS [20] is $o((n + l) + (S + d) + (n + m))$, where n defines task size and l defines tasks length, s defines workload size, d defines workload deadlines, and m time taken to allocate tasks on the virtual machine. The proposed model has attained good performance when compared with the existing models which has been experimentally proven in the next section.

4. Results and discussions

4.1 Experimental setup

In this work, the experimentation has been carried out on Intel-Core i5 processor having 16GB RAM and storage of 1TB HDD. The Montage workload has been used to evaluate the proposed model and other existing models. The existing models considered in this work are EMS [21], and EC-WSC [23]. EMS model main focus was to reduce the energy consumption during the execution of the workloads. They have used a heuristic approach for optimizing the issue of energy consumption. Further, the EC-WSC model main focus was to reduce energy consumption. This work utilized efficient resources for the execution of the workload. All the models EMS, EC-WSC and WCS-ERP were simulated using the CloudSim [25] simulator. The results have been discussed in terms of executional time, average power, average sum, energy consumption and reliability. In the next section, the description of the Pegasus Montage scientific workload.

4.2 Montage workload

Pegasus montage is a scientific workload management system that is used to automate and manage complex scientific workloads. It is specifically designed to support the creation and execution of data-intensive workloads, such as those used in astronomical image processing. Montage is a set of tools that are used to create mosaics of astronomical images. The process of creating a mosaic involves aligning and combining multiple images to create a single seamless image. This process can be time-consuming and computationally intensive, and Montage is designed to automate and optimize this process. More information can be obtained from [26].

4.3 Execution time

Execution time in cloud computing refers to the amount of time it takes for a task or workload to complete in a cloud computing environment. This includes the time it takes to transfer data to and from the cloud, process the data using cloud resources, and return the results to the user. The executional time required for the execution of the montage 25, montage 50, and montage 100 by the existing EMS, and EC-WSC and the proposed WSC-ERP has been evaluated and the results have been given in Fig. 2. Using the Fig. 2, it can be seen that the EMS and EC-WSC takes more time for the execution in comparison to the WSC-ERP. The results show that the WSC-ERP has showed an improvement of 80.27%, and 78.45% for average execution time when compared with the EMS and EC-WSC respectively. The WSC-ERP reduces the time using the Eq. (8). By defining the start and end time for the execution of the workload, the EC-WSC reduces the time required for executing small and large workloads.

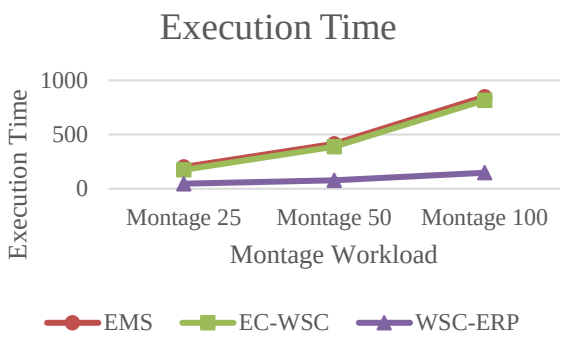

Figure. 2 Execution time

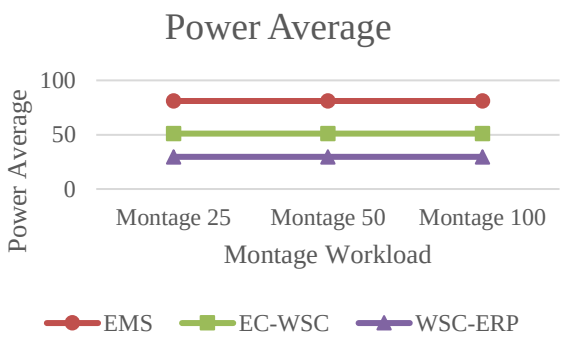

Figure. 4 Power average

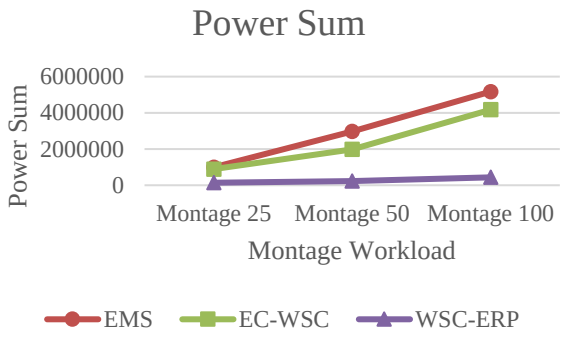

Figure. 3 Power sum

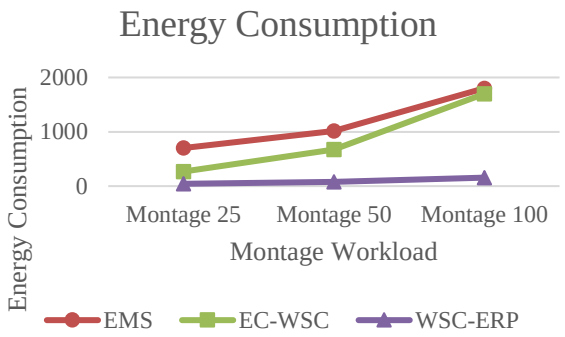

Figure. 5 Energy consumption

4.4 Power sum

In cloud computing, power sum refers to the total power consumption or energy usage of a data center or cloud infrastructure. The power sum for the execution of the montage 25, montage 50, and montage 100 by the existing EMS and EC-WSC and the proposed WSC-ERP has been evaluated and the results have been given in Fig. 3. Using the Fig. 3, it can be seen that the EMS and EC-WSC take consume more power sum in comparison to the WSC-ERP for the execution. The results show that the WSC-ERP showed an improvement of 89.98% and 87.49% for average power sum when compared with the EMS and EC-WSC respectively. The WSC-ERP model reduces the power sum using the constraints t_l, $q_l^{\uparrow}$ and $g_l^{\uparrow}$ presented in Eq. (6).

4.5 Power average

In cloud computing, power average refers to the average power consumption or energy usage of a data center or cloud infrastructure over a given period of time. This metric is often used to estimate the overall power usage and costs of a cloud deployment and to identify potential energy-saving opportunities. The power average for the execution of the montage 25, montage 50, and montage 100 by the existing EMS and EC-WSC and the proposed WSC-ERP has been evaluated and the results have been given in Fig. 4. Using the Fig. 4, it can be seen that the EMS and EC-WSC consume more power average in comparison to the proposed WSC-ERP for the execution. The results show that the WSC-ERP showed an improvement of 63.44% and 41.93% for average power average when compared with the existing EMS and EC-WSC respectively. The WSC-ERP model reduces the power average using the constraints t_l, $q_l^{\uparrow}$ and $g_l^{\uparrow}$ presented in Eq. (6).

4.6 Energy consumption

The energy consumption for the execution of the montage 25, montage 50, and montage 100 by the existing EMS and EC-WSC and the proposed WSC-ERP has been evaluated and the results have been given in Fig. 5. Using the Fig. 5, it can be seen that the EMS and EC-WSC consume more energy in comparison to the proposed WSC-ERP for the execution. The results show that the WSC-ERP showed an improvement of 92.58% and 87.87% for average energy consumption when compared with the existing EMS and EC-WSC respectively. The WSC-ERP model reduces the energy consumption using the constraints t_l, $q_l^{\uparrow}$ and $g_l^{\uparrow}$ presented in Eq. (6).

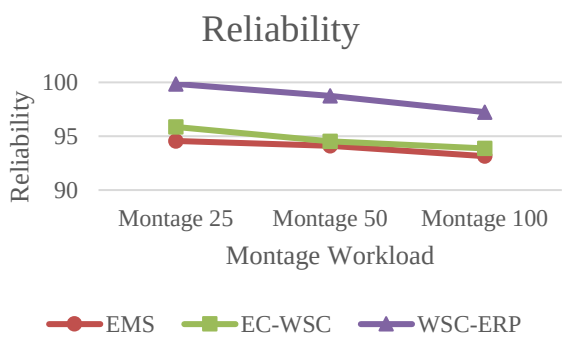

Figure. 6 Reliability

Table 2. Comparative study

	EMS [21], 2022	EC-WSC [23] ,2023	WSC-ERP (Proposed)
Heterogeneous computing	Yes	Yes	Yes
Workload type	Complex	Complex	Complex
Workload size	Small to large	Small to large	Small to large
QoS Metrics	Energy	Processing Time and Energy	Energy, Processing time, resource utilization, & deadlines
Optimization strategy	Heuristic	Heuristic	Heuristic
Replanning	No	No	Yes
Reliability	Yes	No	Yes
Availability	Yes	Yes	Yes

4.7 Reliability

Reliability is a critical aspect of cloud computing as users rely on cloud services to be available and perform as expected. Reliability refers to the ability of a cloud system to deliver its services consistently and without interruption, even in the face of hardware failures, software errors, and network disruptions. The reliability for the execution of the montage 25, montage 50, and montage 100 by the existing EMS and EC-WSC and the proposed WSC-ERP has been evaluated and the results have been given in Fig. 6. Using the Fig. 6, it can be seen that the EMS and EC-WSC are less reliable in comparison to the proposed WSC-ERP for the execution. The results show that the WSC-ERP showed an improvement of 4.97% and 4.07% for average power average when compared with the existing EMS and EC-WSC respectively. The WSC-ERP model has reduced the execution time and energy consumption as shown in the previous sections. Also, this work reduces the time complexity using the Eq. (10), hence, the WSC-ERP is more reliable in comparison to the EMS and EC-WSC models.

4.8 Comparative study

The Table 2 presents a comparative study of the existing works and the proposed works. All three systems employ heterogeneous computing, utilizing multiple types of processors with different capabilities. The workload in each system is complex and can range from small to large tasks, requiring significant computational resources. Quality of service (QoS) metrics differ across the systems, with EMS [21] focusing on energy consumption, EC-WSC [23] on processing time and energy, and WSC-ERP considering energy, processing time, resource utilization, and deadlines. The optimization strategy for all three systems is heuristic-based. Replanning is not included in EMS [21] and EC-WSC [23], but WSC-ERP incorporates it, enabling dynamic adjustments to changing workloads. Reliability is present in EMS [21] and WSC-ERP, while EC-WSC [23] lacks reliability. However, all three systems offer availability features, ensuring continuous access and operation.

5. Conclusion

This paper proposes a novel approach to cloud workload task scheduling in web service composition that incorporates efficient replanning and time complexity analysis. The proposed approach takes into account various factors, such as task dependencies, and resource availability, to generate an optimized schedule that minimizes execution time and maximizes resource utilization. To achieve efficient replanning, the proposed approach uses a dynamic programming-based algorithm that can quickly adapt to changes in the system, such as resource failures or changes in user preferences. The approach also incorporates time complexity analysis to ensure that the generated schedule is feasible and can be executed within the available time and resource constraints which the previous existing works have not addressed. The proposed approach is evaluated using a Montage scientific workload and compared with existing model. The results show that the proposed approach is effective in generating optimized schedules with efficient replanning and low time complexity. The approach can be applied to a wide range of web service composition scenarios and can help improve the efficiency and reliability of

cloud workload task scheduling. Overall, the proposed approach provides a promising solution to the complex task of cloud workload task scheduling in web service composition, with efficient replanning and time complexity analysis as key components. Lastly, for the future work, security and compliance can be provided during the scheduling of workloads for web server composition in the cloud.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Kavita D. Hanabaratti and guide Rudragoud Patil collaborated on the research project focusing on the "Task Replanning Algorithm for Web-Service Provisioning." Kavita D. Hanabaratti took the lead and made significant contributions throughout the entire work. She conducted the research, designed and implemented the Task Replanning Algorithm, performed experiments, and analyzed the results. Kavita also prepared the initial draft of the manuscript. Rudragoud Patil provided valuable assistance and mentorship to Kavita throughout the research process. He offered guidance in formulating the research objectives, refining the algorithm design, and reviewing the experimental setup. Rudragoud Patil played a crucial role in providing technical expertise and critical feedback, which helped in strengthening the research findings and the overall quality of the paper.

References

[1] M. U. Sana, and Z. Li, "Efficiency aware scheduling techniques in cloud computing: a descriptive literature review", *PeerJ Computer Science*, Vol. 7, p. 509, 2021.

[2] R. Han, C. H. Liu, Z. Zong, L. Y. Chen, W. Liu, S. Wang, and J. Zhan, "Workload-Adaptive Configuration Tuning for Hierarchical Cloud Schedulers", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 30, No. 12, pp. 2879-2895, 2019.

[3] S. M. Kumar, P. Senthil, E. Sabitha, P. J. S. Kumar, A. Balasundaram, and S. Ashokkumar, "Pre-emptive Approach for Resource Scheduling in Cloud Computing", In: *Proc. of 2020 International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, pp. 276-281, 2020.

[4] S. R. Sultana, K. S. Meena, P. Alaguvathana, and K. Abinaya, "Priority based Scheduling Algorithm using Divisible Load Theory in Cloud", In: *Proc. of 2020 Fourth International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, India, pp. 771-776, 2020.

[5] S. A. Murad, A. J. Muzahid, Z. R. Azmi, M. I. Hoque, and M. Kowsher, "A review on job scheduling technique in cloud computing and priority rule based intelligent framework", *Journal of King Saud University Computer and Information Sciences*, Vol. 34, No. 6, pp. 2309–2331, 2022.

[6] A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga, "Resource Management Techniques for Cloud/Fog and Edge Computing: An Evaluation Framework and Classification", *Sensors*, Vol. 21, p. 1832, 2021.

[7] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers: a survey on software technologies", *Cluster Computing*, Vol. 26, pp. 1845-1875, 2023.

[8] V. Mohammadian, N. J. Navimipour, M. Hosseinzadeh, and A. Darwesh, "Fault-Tolerant Load Balancing in Cloud Computing: A Systematic Literature Review", *IEEE Access*, Vol. 10, pp. 12714-12731, 2022.

[9] S. Afzal and G. Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification", *Journal of Cloud Computing*, Vol. 8, No. 22, 2019.

[10] F. Tapia, M. A. Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis, "From Monolithic Systems to Microservices: A Comparative Study of Performance", *Applied Science*, Vol. 10, p. 5797, 2020.

[11] Z. Mohtajollah and F. Adibnia, "A Novel Parallel Jobs Scheduling Algorithm in The Cloud Computing", In: *Proc. of 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, pp. 243-248, 2019.

[12] G. T. Andreazi, J. C. Estrella, S. M. Bruschi, R. Immich, D. Guidoni, L. A. P. Júnior, and R. I. Meneguette, "MoHRiPA-An Architecture for Hybrid Resources Management of Private Cloud Environments", *Sensors*, Vol. 15, No. 21, p. 6857, 2021.

[13] L. Yang, Y. Xia, L. Ye, R. Gao, and Y. Zhan, "A Fully Hybrid Algorithm for Deadline Constrained Workflow Scheduling in Clouds", *IEEE Transactions on Cloud Computing*, pp. 1-15, 2023.

[14] Z. Li, H. Yu, G. Fan, and J. Zhang, "Cost-efficient Fault-tolerant Workflow Scheduling for Deadline-constrained Microservice-based Applications in Clouds", *IEEE Transactions on*

Network and Service Management, pp. 1-1, 2023.

[15] T. Hai, J. Zhou, D. Jawawi, D. Wang, U. Oduah, C. Biamba, and S. K. Jain, "Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes", *Journal of Cloud Computing*, Vol. 12, No. 15, 2023.

[16] H. Tarawneh, I. Alhadid, S. Khwaldeh, S. Afaneh, "An Intelligent Cloud Service Composition Optimization Using Spider Monkey and Multistage Forward Search Algorithms", *Symmetry*, Vol. 14, No. 28, 2022.

[17] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing", *Journal of Cloud Computing*, Vol. 12, No. 24, 2023.

[18] 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.

[19] M. Kumar, A. Kishor, J. Abawajy, P. Agarwal, A. Singh, and A. Zomaya, "ARPS: An Autonomic Resource Provisioning and Scheduling Framework for Cloud Platforms", *IEEE Transactions on Sustainable Computing*, Vol. 7, No. 2, pp. 386-399, 2022.

[20] J. K. Konjaang and L. Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing", *Journal of Cloud Computing*, Vol. 10, No. 11, 2021.

[21] 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, 2021.

[22] S. Tao, Y. Xia, L. Ye, C. Yan and R. Gao, "DB-ACO: A Deadline-Budget Constrained Ant Colony Optimization for Workflow Scheduling in Clouds", *IEEE Transactions on Automation Science and Engineering*, pp. 1-16, 2023.

[23] K. D. Hanabaratti and S. F. Rodd, "Evolutionary Computing based Web Service Composition Technique for Scheduling of Workload under Cloud Environment", *Indian Journal of Science and Technology*, Vol. 15, No. 2, pp. 69-80, 2022.

[24] K. D. Hanabaratti and R. Patil, "Efficient algorithm for replanning web service composition", *The Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, Vol. 31, No. 1, 2023.

[25] M. O. Ahmad and R. Z. Khan, "Cloud Computing Modeling and Simulation using Cloud Sim Environment", *International Journal of Recent Technology and Engineering*, Vol. 8, No. 2, pp. 5439–5445, Jul. 2019.

[26] Workflow gallery, Pegasus WMS. Available at: https://pegasus.isi.edu/workflow_gallery/gallery/montage/index.php (Accessed: May 2, 2023).