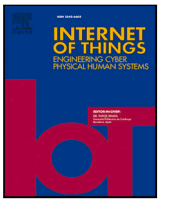
Internet of Things 23 (2023) 100909

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

A fault tolerance aware green IoT workflow scheduling algorithm for multi-dimensional resource utilization in sustainable cloud computing

Mustafa Ibrahim Khaleel

University of Sulaimani, College of Science, Computer Department, Sulaimani, Kurdistan Regional Government 46001, Iraq

ARTICLE INFO

Keywords:
Workflow placement
Service profit
Fault tolerance
Increased reliability

ABSTRACT

In recent times, the distributed cloud computing landscape has witnessed a remarkable surge in processing vast amounts of data and the crucial need to maintain service level agreements (SLAs) between providers and consumers. In this dynamic environment, cloud resources are inherently multidimensional, encompassing computing machines and communication connections susceptible to failures and energy-related considerations. However, given two-objective energy and reliability optimization, existing time allocation policies focus primarily on optimizing a single intent, which leads to system degradation in environments that generate problematic constraints from executable processing units. To handle the shortcomings of the application placement policies, we suggest a solution in the form of a three-phase bi-objective workflow scheduling issue called (Bi-OWSP). This three-phase approach aims to optimize workflow scheduling by simultaneously considering energy and reliability as two vital objectives. To achieve this, we employ two distinct algorithms. First is the stepwise dynamic voltage and frequency scale algorithm ensures energy efficiency by calculating optimal frequencies, reducing energy usage, and minimizing task mapping time. The second algorithm is the reliability-conscious heterogeneous fault tolerance approach, which emphasizes avoiding high-deficit servers and communication links to enhance system reliability. Furthermore, we introduce an energy-aware stepwise reliability maximization algorithm, which intelligently selects the best combination of task-server pairs to achieve energy minimization and reliability maximization. Through extensive simulation experiments on artificial and real-world workflow applications, we demonstrate the significance of Bi-OWSP in providing superior energy-reliability compensation solutions compared to competing algorithms.

1. Introduction

As a well-established model, cloud computing has drawn considerable attention to ubiquitous, mobile, and cognitive computing; distinctive computing features such as mobility, accessibility, and intelligence are delivered to the users [1]. The scalable and elastic cloud resources provide more intelligent services pervasively based on the dynamic environment [2]. However, the ever-increasing demand and low capability of front-end computing machines require a robust back-end paradigm to effectively execute complex computational data requests. This large-scale data processing results in tremendous energy expenditure in cloud DCs [3].

Amazon's analysis indicates that data centers' energy costs make up around 42% of their total operational expenses. This straight influences the QoS and reliability of incoming task execution. In addition, these operating expenses could potentially jeopardize the

E-mail address: mustafa.khaleel@univsul.edu.iq.

https://doi.org/10.1016/j.iot.2023.100909
Received 11 June 2023; Received in revised form 12 August 2023; Accepted 12 August 2023
Available online 16 August 2023

ability of cloud providers to offer timely services due to their increased costs [4]. These challenges are becoming a primary research problem: Increasing resource utilization through efficient use of cloud resources requires a rigorous analysis effort focusing on developing and implementing intelligent cloud mapping solutions to deliver the best outcomes in terms of profit, reliability, and QoS-satisfactory applications at a low cost of power. Placing applications on processing machines with fine granularity (i.e., low failure rate) is a promising solution for increasing resource utilization and improving service quality. Though machine resources are sharable by numerous applications, selecting the best fitting machine to accommodate newly assigned tasks or to integrate them with existing applications is a complex task, and it has become a commonly discussed topic in recent years [5]. Determining where to place scheduled task applications involves packing multi-dimensional vectors; no precise polynomial solution exists. An adequate placement policy can have tri-objectives of: (i) *minimizing energy consumption and making data centers greener*, (ii) *maximizing the application placement reliability*, and (iii) *reducing resource wastage and escalating the provider's profit.*

The application placement process entails managing diverse services across different computing devices, including memory allocation, processor usage, disk utilization, and network bandwidth assignment. As the frequency of devices increases, so does the complexity of managing these multi-dimensional resources. Therefore, minimizing the frequency of active processing machines becomes crucial. Achieving this goal involves optimizing machine CPU frequency and resource utilization while considering machine and communication link failures. Despite efforts to address this optimization problem, current approaches lack consideration for a distributed application placement system. Instead, they rely on central application placement policies that require powerful servers to collect all mapping strategies and make decisions. This centralized approach has limitations, as it increases the risk of a single point failure and raises storage, transmission, and computing costs. Therefore, the scaling issue of application placement requires a distributed application placement algorithm to distribute the decision-making process of application placement strategies (i.e., energy-reliability-aware mode) among various processing machines and strives to achieve a resource-balanced manner.

In heterogeneous clusters with dynamically variable voltages and speeds, optimizing reliability and energy saving to address priority-constrained tasks in combination with optimization is addressed in this paper. Our application placement scheme mainly deals with three non-trivial sub-problems: prioritization restrictions, energy savings, and optimum reliability.

- *Precedence Constraining*. In contrast to standalone application packages, the presence of priority constraints in parallel applications adds a significant level of complexity to the design and analysis of heuristic placement algorithms.
- *Energy Conserving*. To achieve minimal energy consumption during the application placement process, the CPU frequency should be adjustable to accommodate various speeds with only minor performance losses.
- *Reliability Maximization*. To attain the highest level of reliability, it is essential to have a suitable task-server pair for each scheduling cycle.

Addressing the abovementioned sub-problems is crucial to enhancing the Profit-to-Utilization Ratio (PUR). This paper has modified three algorithms, namely EDVFS, RHFT, and ERMax, to achieve this goal. The EDVFS algorithm assesses the posterior state of each feasible processing machine to determine its suitability for hosting assigned tasks. It trains the server hosts for different utilizations, regulates CPU frequencies, and identifies optimal and preferred frequency ranges. On the other hand, the RHFT and ERMax algorithms focus on ensuring robust failure tolerance for application placement. They refrain from using server machines with high deficits and communication links to enhance reliability. The task server is selected based on the most appropriate combination while considering the trade-off between energy efficiency and reliability.

The significant **contributions** of the paper include.

- A new approach to partitioning a multi-dimensional space is proposed, aiming to strike a harmonious balance between maximizing profits and efficiently utilizing available resources, with a specific emphasis on enhancing fault tolerance.
- While conducting the application placement process, the detection of single-point failures is performed to avoid the presence of unused resources and improve both energy cost efficiency and task placement reliability.
- Additional reliability analysis is conducted to attain an optimal reliability ratio for application placement. This analysis involves utilizing the Poisson distribution to assess the hazard rate of task execution within the underlying cloud clusters. The main objective is to identify and implement the most effective solutions for placing applications, with a particular focus on prioritizing reliability.

2. Related work

Gupta et al. [6] created resource utilization factors to improve the use of physical machines during the virtual machine installation process. The objective of their algorithm was to decrease the following: (i) *the number of devices that are underloaded using a multi-dimensional resource usage framework*, (ii) *the number of currently utilized machines to enhance the efficiency of utilization usage*, and (iii) *the VM migrations to lower resource waste, energy consumption, and degradation of SLA*.

The work of [7] takes into account the time of delivery of services, access to opportunities, and internal communication delays. Furthermore, they designed decentralized organizational systems to replace and disseminate components to overcome the limits of administrative overhead, a single failure point, and redundant communication. Nabavi et al. [8] optimize the placement schemes of multiple objective virtual machines in a traffic-sensitive way. Through the execution of artificial bee colony optimization algorithms, the traffic forms between VMs are aligned with the distances of communication between them, aimed at minimizing network traffic between virtual machines and reducing DC's power expenses.

Khan [9] has developed a strategy for VM consolidation that uses normalization to bring VMs online and reduce energy consumption and SLA degradations. The primary aim of the algorithm proposed in the study is to decrease the numeral of VM migrations since too many of these migrations can negatively impact QoS through increased overhead during runtime. To identify overloaded machines, resource parameters were included, and the CATR was computed to specify underutilized devices. Additionally, the algorithm employs a standardized resource parameter for both physical and VMs to determine the most efficient host for migrating a VM.

Zhou et al. [10] addressed the challenge of distributing harvested energy to machines powered by renewable sources and assigning workloads on-site. Their goal was to enhance the performance of individual equipment and the overall system while operating within energy and reliability limitations. The energy distribution was regulated based on the system's state. To manage low-energy systems, the authors proposed a game theory approach, where energy allocation was viewed as a collaborative game involving multiple machines, and Nash's bargaining solution was utilized. To satisfy reliability constraints, they studied the reliability-optimality of tasks scheduled on various devices and created an analysis-based, reliability-aware task allocation paradigm.

Ramzanpoor et al. [11] presented a heuristic approach to tackle a single failure point and improve the reliability of applications against failure. Their algorithm reduces bandwidth waste in relation to the dependence on application components. Deploying components for an application is an optimization problem involving multiple objectives. Optimization aims to lower energy consumption and decrease total latency among the different parts of the application.

Swain et al. [12] have studied the solution of two extraordinary possibilities for the problem previously mentioned. There are two aspects to consider. The first aspect concerns tasks with an anticipated completion time on a machine with an equivalent failure rate. The second aspect pertains to the application with a corresponding execution time.

Kanwal et al. [13] have developed a multi-phase error tolerance paradigm based on genetic algorithms for putting applications on appropriate computing machines. Their heuristics enable the optimal mapping of VMs to cloud users, taking into account SLAs, in an efficient manner. It comprises four main steps: the individual step, local step, global step, and fault tolerance step. In the first step, the local capability of each user is calculated, and the global capacity of multiple users is estimated based on the SLA of the global capacity of the fitness stage.

Rani et al. [14] have developed a double-objective optimization problem that minimizes energy costs and maximizes the reliability of the application execution. They first designed a reliable green workflow schedule algorithm based on ε-fuzzy dominance in cloud infrastructures (FDRGS). The best solution is formulated by assessing the false dominance value of the equilibrium solution (relative adequacy). Then, they employed DVFS techniques to further decrease energy expenses by controlling the CPU frequency.

Medara et al. [15] have presented a framework for multi-objective workflow positioning to formulate energy reliability efficiency in the cloud network. Five sub-algorithms are assessed in their paradigm. First, they propose task-order calculation algorithms to maintain task dependencies. Then, they developed task clustering algorithms to reduce communications expenses and energy costs. They then developed an algorithm for the automatic distribution of sub-target time to determine the sub-makespan of the application.

Hassan et al. [16] proposed SERAS paradigm. It operates on the basis of the DVFS strategy for applications executed in cloud environments. It undermines the deadline for the workflow of the application. In addition, the DVFS method reduces the CPU frequency of the machine without violating application time limits. The SERAS algorithm is designed to ensure that the application is hosted on servers with appropriate frequency levels while maintaining the reliability of cloud-based hardware.

Garge et al. [17] suggested a novel application placement scheme to adjust the reliability of the execution process and energy costs within the QoS requirements specified by the user. The proposed method operates by following four key steps: First, calculating the priority of each application based on its energy and reliability needs—second, grouping applications with similar energy and reliability requirements into clusters. Thirdly, the target processing time among the clusters is distributed based on their priority. They were, finally, managed each group to cope with the most suitable operational frequency.

Yao et al. [18] presented an optimization method that effectively balances handling time reliability and operating costs. Not only does it achieve optimal balance, but it also provides efficient solutions to both factors. They designed an integer linear programming (ILP) problem to minimize operating costs while maximizing system reliability.

Li et al. [19] invented an intelligent approach for scheduling cloud resources to address resource failure. Their method, called Real-time and dynamic Fault-tolerant Scheduling (ReadyFS), efficiently maps scientific workflow applications, optimizing running time deadlines and enhancing resource usage despite resource failures. The solution handles both host permanent failure (HPF) and virtual machine temporary failure (VMTF) through two techniques: replication with delay execution (RDE) and checkpointing with delay execution (CDE). They also improve the reliability model by considering the host transient failure (HTF) and propose the rescheduling (ReSC) model to address it. To manage resource demands and improve the efficiency of allocated resources, they employ a resource adjustment (RA) strategy.

In their article, Lin, W et al. [20] put forward dual IoT-aware multi-resource job mapping strategies for cloud networks with heterogeneous resources: central resource load balancing and time balancing. The primary objective is achieving better load balancing, fulfilling SLA requirements, reducing task implementation time, and optimizing energy consumption. The proposed solution aims to identify the most efficient combination of tasks and their corresponding destinations (i.e., VMs) to enhance the overall system's throughput. Incoming jobs are queued and dispatched sequentially while waiting for appropriate VMs. The selection of virtual machines is based on relative load or relative time cost considerations.

The problem of insufficient VM placement causing resource fragmentation was addressed by Li et al. [21]. They aimed to consolidate more VMs onto fewer computing hosts while ensuring proper management to avoid negative impacts on system efficiency. Li et al. [21] focused on enhancing energy consumption and service quality by optimizing the VM placement policy. To achieve this, they used a discrete differential evolution algorithm to search for the global optimal VM placement. Additionally,

Table 1
A synopsis of the research conducted on the topic [6–18,22].

References	Experimental constraints					Achievement	Limitations
	Fault tolerant	Energy conscious	Reliability conscious	ER-Ratio trade-off	PU-Ratio conscious		
[6]	○	●	○	○	○	The purpose of a MOVMP is to reduce energy costs by lowering the usage of currently active machines	Considering a static placing model and training only the CPU performance metric
[7]	○	●	○	○	○	Deployed time-sensitive applications in the proximity of source data	Failure to take into account the interdependent relationships in the distribution process
[8]	○	●	○	○	○	A multi-objective VM is proposed to optimize energy cost based on ABC optimization method	Certain requirements including node and network link failure rates are disregarded
[9]	○	●	○	○	○	The development of a NVMC model that can efficiently place VMs in real-time while minimizing energy expenditure, breaches of SLAs, and the need for VM migrations	Their proposed paradigm does not consider dependency challenges between components
[10]	○	●	●	○	○	A game theoretic approach and reliability-aware task assignment heuristic are proposed to optimize energy and reliability	The proposed approach does not provide details on how to distribute components when faced with a single point of failure
[11]	●	●	●	○	○	A MOCSA model is presented to solve the conflict objectives of minimizing the energy cost and latency between pair of applications	Current cloud networks do not take into account network failures, and the limited resources of cloud nodes make them more susceptible to failure
[12]	○	○	●	○	○	A dual-phase heuristic approach is proposed for task ordering and mapping to the most suitable machine	The approach does not ensure a perfect deployment solution for the problem of a single pinpoint of failure
[13]	●	○	●	○	○	A genetic algorithm-based multi-phase fault tolerance heuristic is proposed for task placement that can reduce the failure rates of the mapping tasks	Failing to restart set-up virtual machines leads to higher latency and energy outlay
[14]	○	●	●	○	○	The proposal is for a fuzzy dominance paradigm that bargains both the dependability and energy charge of the incoming tasks	The proposed heuristic method encounters significant transmission indecisiveness when users are distant from the source units
[15]	○	●	●	○	○	The proposal is for a multi-aiming DAG dispatching approach that manage the efficiency of energy and reliability through incoming requests' dependencies	When there is an immense volume of incoming user propositions, latency and communication wait are significant
[16]	○	●	●	○	○	The suggestion is for a new mapping heuristic that considers both energy costs and reliability, and sums the optimal CPU frequency to achieve the best possible reliability and finish time	The proposed algorithm has a lack of leakage prevention, which results in inadequate energy conserving

(*continued on next page*)

they employed various approaches, such as detecting overutilized and underutilized servers and selecting VMs to supplement their solution.

Table 1 provides a summary of the research conducted on the topic. It is important to consider this trade-off carefully when making decisions about where to place applications.

Table 1 (*continued*).

References	Experimental constraints					Achievement	Limitations
	Fault tolerant	Energy conscious	Reliability conscious	ER-Ratio trade-off	PU-Ratio conscious		
[17]	○	●	●	○	○	Composing both the dependability of tasks and energy outlay while also adhering to the QoS constraints	The specific obstacles to energy cost are taken into account, but hindrances such as PUR, time request, and EQA are neglected
[18]	○	●	●	○	○	A trade-off reliability and operational outlay is crafted to maximize reliability and minimize the operating cost of nodes	Existing work lacks for mobility, and their approach is not scalable for the hugeness of online appliances
[22]	●	○	●	○	○	The proposal of a cost-efficient workflow mapping paradigm that is fault-tolerant, aiming to minimize the cost of executing requests and the overall time taken while ensuring their dependability	The PUR approach is not taken into account during the placement of cloud apparatuses

Table 2
The primary symbols and representations outlined in this paper.

Symbols	Representations Meanings	Applied in Formula
$\mathcal{W}\tau_s$	The start time slot in windows $\mathcal{W}\tau$	Eq. (1)
$\mathcal{W}\tau_e$	The end time slot in windows $\mathcal{W}\tau$	Eq. (1)
$\mathcal{W}^i$	The ith scheduled workflow on server instances	Eq. (3)
$EC(\mathcal{W}^i)$	The energy expenses of executing a workflow $\mathcal{W}^i$	Eqs. (3), (8), (9), (26)
$\mathcal{RC}(\mathcal{W}^i)$	The reliability outcome of executing a workflow $\mathcal{W}^i$	Eq. (3)
$\mathcal{P}_{C_j}$	The independent power cost of the servers $\mathcal{S}_j$	Eqs. (5), (6), (8), (23), (24)
C_e	The effective loading capacitance of the servers' CPU	Eqs. (6), (7), (8)
$\mathcal{V}_{dd}^2$	The supply voltage of the servers' CPU	Eq. (7)
$EC(\mathcal{S}_j^f)$	The energy expense of the node $\mathcal{S}_j$ at frequency f	Eqs. (8), (20)
$\mathcal{R}(\mathcal{L}_{i,j}^{\tau}, \Delta\tau)$	The reliability of communication link the $\mathcal{L}_{i,j}$ during $\Delta\tau$	Eq. (10)
$\mathcal{RE}(\tau)$	The reliability enhanced of the task τ	Eq. (11)
$\mathcal{D}_{i,j}$	The minimum delay between servers $\mathcal{S}_i$ and $\mathcal{S}_j$	Eq. (13)
$\mathcal{B}_{i,j}$	The minimum bandwidth between servers $\mathcal{S}_i$ and $\mathcal{S}_j$	Eqs. (13), (21)
$F_{i,j}$	The communication failure rate between servers $\mathcal{S}_i$ and $\mathcal{S}_j$	Eqs. (13), (21)
$suc(\tau_i)$	The summed number of successors applications of the task τ_i	Eq. (15)
$\mathcal{S}_j^f(\mathcal{OPT})$	The optimal frequency of the servers $\mathcal{S}_j$	Eq. (16)
$e\mathcal{C}\tau$	The earliest completion time of workflow $\mathcal{W}^i$	Eq. (17)
$\mathcal{C}\tau\left[\mathcal{S}_j^f(\tau_i)\right]$	The computing time of task τ_i on $\mathcal{S}_j$	Eqs. (17), (18), (19), (21)
$EC\left(\mathcal{S}_j^f(min)\right)$	The energy cost of $\mathcal{S}_j$ at its minimum frequency	Eq. (20)
$EC\left(\mathcal{S}_j^f(max)\right)$	The energy cost of $\mathcal{S}_j$ at its maximum frequency	Eq. (20)
$EC\left(\mathcal{S}_j^f(\tau_i)\right)$	The energy cost of $\mathcal{S}_j$ at its best frequency	Eq. (20)
$\mathcal{R}\left(\mathcal{S}_j^f(min)\right)$	The reliability cost of $\mathcal{S}_j$ at its minimum frequency	Eq. (20)
$\mathcal{R}\left(\mathcal{S}_j^f(max)\right)$	The reliability cost of $\mathcal{S}_j$ at its maximum frequency	Eq. (20)
$\mathcal{R}\left(\mathcal{S}_j^f(\tau_i)\right)$	The reliability cost of $\mathcal{S}_j$ at its best frequency	Eq. (20)
$pre(\tau_i)$	The predecessors jobs linked to the task τ_i	Eqs. (21), (23)
$\mathcal{N}_{F\tau}^i$	The task τ's updated time in the n$\mathcal{CP}$ queue	Eq. (21)
$F_{upst_{i,j}}$	The failure rate between two upstreaming servers $\mathcal{S}_i$ and $\mathcal{S}_j$	Eq. (21)
$\mathcal{B}_{upst_{i,j}}$	The bandwidth between two upstreaming servers $\mathcal{S}_i$ and $\mathcal{S}_j$	Eq. (21)

3. Problem statement

This part assesses the reliability of where applications are placed and how energy is utilized. We also present analytical models for this purpose. To simplify, we have included Table 2 that outlines the specific terminology used in our study.

3.1. Problem definition

In our proposed model, we consider multiple clusters $\mathcal{C} = \{C_1, C_2, C_3, \ldots C_z\}$ in $\mathcal{D}_k$. A batch of processing servers, $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \ldots \mathcal{S}_n\}$ are defined. Each server has a group of virtual machines $\mathcal{V}m = \{\mathcal{V}m_1, \mathcal{V}m_2, \mathcal{V}m_3, \ldots \mathcal{V}m_m\}$. Each server is characterized by multiple-dimensional resources such as power consumption, execution reliability, network bandwidth, and memory.

The users of our system model submit tasks $T = \{\tau_1, \tau_2, \tau_3, \ldots \tau_q\}$ to the servers in the form of single tasks, pipelines, or complex workflows framed in DAG to improve energy and reliability costs by placing them in efficient VMs; this process is known as *application placement scheme (APS)*.

3.2. Problem formulation

This section describes the system for placing applications, which is founded on the subsequent assumptions and hypotheses.

$$\mathcal{Y} = \sum_{i=1}^{q}\sum_{j=1}^{n}\left[S_j, \mathcal{SL}_{i,j}\right]_{\mathcal{W}\tau_s, \mathcal{W}\tau_e} \tag{1}$$

$\mathcal{SL}_{i,j}$ is the schedule length of placing modules on the S_j, i.e., the aggregated duration for computing tasks in the time window $\mathcal{W}\tau[s-e]$. Furthermore, we define the capacity C of the server S_j as C_j^d. The d-dimensional resource is subject to the resource limitations.

$$\left[S_j, \mathcal{SL}_{i,j}\right]_{\mathcal{W}\tau_s, \mathcal{W}\tau_e} = \min\left[S_j, \mathcal{SL}_{i,j}\right]_{\mathcal{W}\tau_s, \mathcal{W}\tau_e}$$
$$1 \le d \le \mathcal{D}$$
$$C_j^d \ge \sum_{i=1}^{\delta} \mathcal{P}_{s,\upsilon}^{i} \times \mathcal{R}e_{s,\upsilon}^{d}$$

The context includes: $\mathcal{R}e_{s,\upsilon}^{d}$ is the $\mathcal{S}$the dimensional resource of the υth VM and $\mathcal{P}_{s,\upsilon}^{i}$ is a matrix which its components are defines according to Eq. (2).

$$\mathcal{P}_{s,\upsilon}^{i} = \begin{cases} 1, & \text{if the } \upsilon\text{th is dispatched on the } \mathcal{S}\text{th server;} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Two dimensions of resources is tested: server energy consumption and application reliability. In other words, $d = 2$. The objective is to address the issue of scheduling with multiple goals of reducing energy costs and maximize dependability. Reduced active server number has proven to reduce energy costs and improve reliability. Therefore, we define our application placement problem as follows.

Definition 1. Given with:

- *A bag of parallel tasks $T = \{\tau_1, \tau_2, \tau_3, \ldots \tau_\upsilon\}$, where a task τ_i is characterized by: $(\alpha_i, e_i, d_i, \omega_i)$ that indicates the task's placing time, processing time, deadline, and weight.*
- *A heterogeneous faulty network environment, $\mathcal{S} = (\mathcal{N}, \mathcal{L})$, where failures of servers and transfer links are unavoidable.*

The goal is to assign tasks on fewer active servers that prioritize energy efficiency and reliability, leading to increased system productivity.

$$\begin{aligned} &\min_{\forall \text{possible mapping}} \left(\mathcal{EC}_{\forall n_j \in \mathcal{S}}(\mathcal{W}^i)\right) \\ &\max_{\forall \text{possible mapping}} \left(\mathcal{RC}_{\forall n_j \in \mathcal{S}}(\mathcal{W}^i)\right) \end{aligned} \tag{3}$$

subject to the following constraints,

$$\forall \tau_i \in T, \forall n_j \in \mathcal{S}; x_{i,j} \in \{0,1\}$$
$$\forall n_j \in \mathcal{S}; \sum_{\mathcal{K}=1}^{\mathcal{P}} x_{i,j} = 1$$

4. The key design of $\mathcal{Bi}-\mathcal{OWSP}$

4.1. Framework of the system

Fig. 1 illustrates the system framework that is being proposed. The system has three primary layers.

a. **Application Layer:** We define workflow applications as a collection of interdependent tasks modeled as a Directed Acyclic Graph (DAG), denoted by $T = (\mathcal{G}, \mathbb{E})$. Here, $\mathcal{G}$ represents the number of interdependent tasks, and $\mathbb{E}$ represents the set of edges, i.e., directed arcs connecting the interdependent tasks. The weight $\omega_{i,j}$ along the edge $\mathbb{E}_{i,j}$ signifies the data size transferred from task τ_i to τ_j. A task is executed when it receives data input from its predecessor tasks, ensuring that the required resources are available. In this approach, the complexity of incoming requests is expressed as a function denoted by $\zeta_i(.)$, which depends on the absolute size of the assigned jobs represented by δ_i. Thus, $\zeta_i(\delta_i)$ consider the computational conditions' relative complexity for the task τ_i. Furthermore, we define $pred(\tau_i)$ and $succ(\tau_i)$ as the predecessor and successor tasks of the task τ_i, respectively.

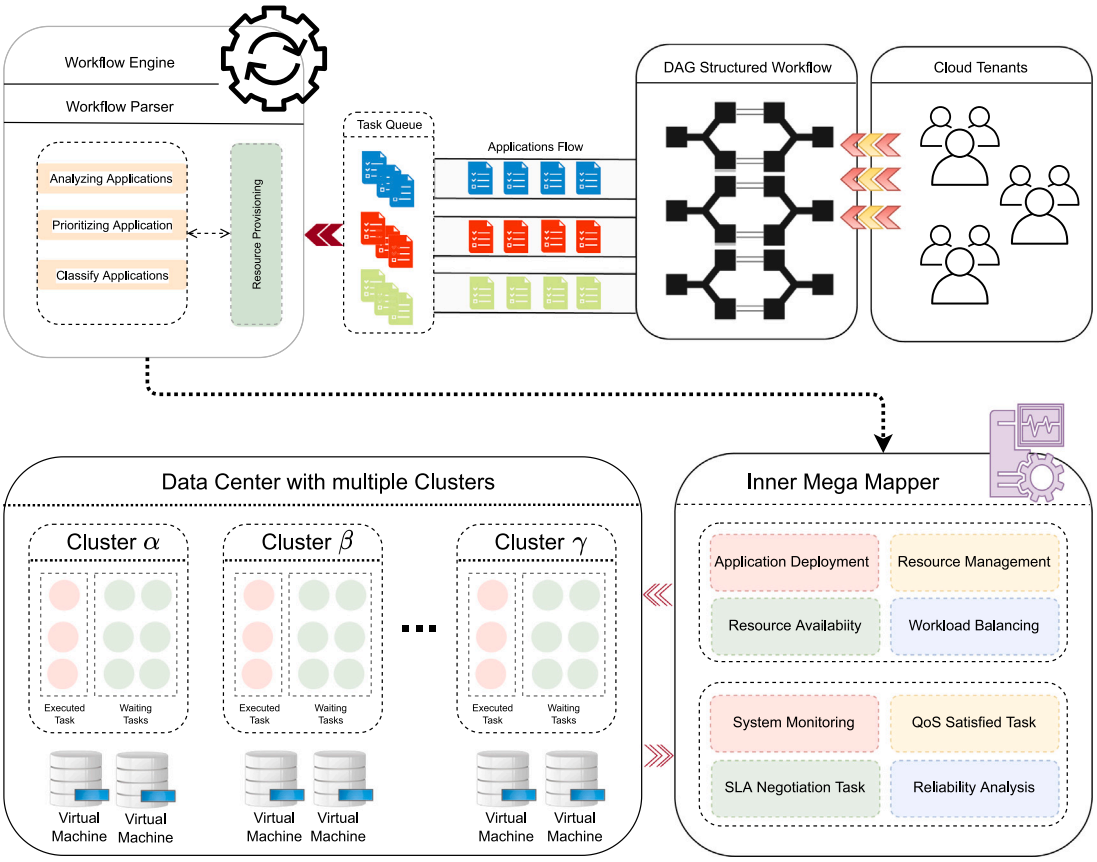


Fig. 1. The cloud system scheduling architecture.

b. **Mega Mapper Layer:** The primary model responsible for application deployment on cloud servers is Mega Mapper. The effective management of resources between cloud service providers and users is of utmost importance. The deployment of applications heavily relies on various parameters, including resource availability, workload balancing, and meeting quality of service (QoS) requirements. These factors are crucial in ensuring the successful execution of assigned tasks and maintaining their dependencies. The architecture of Mega Mapper includes vital components such as a task queue, SLA negotiation factor, reliability analysis, end-to-end delay (EED), and task dispatch. When users submit tasks, they are placed in the task queue, waiting until the Mega Mapper model negotiates the Service Level Agreement (SLA).

c. **Surface Layer:** We define the network system at the core of our study as a weighted network graph, denoted by $\mathcal{S} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ represents a set of directed cloud servers and $\mathcal{L}$ denotes the communication links connecting all servers. Each processing server is characterized by its power consumption $\mathcal{P}_j$ and failure rate F_j. Furthermore, each link connecting a pair of servers is characterized by three parameters: (i) transport bandwidth $\mathcal{B}_{i,j}$, (ii) link failure rate $\mathcal{L}F_{i,j}$, and (iii) minimum link delay $\mathcal{D}_{i,j}$. To address the problem, we formulate a placement matrix denoted by $x_{i,j}$.

$$x_{i,j} = \begin{cases} 1, & \text{if } \tau_i \rightarrow \mathcal{S}_j^v; \\ 0, & \text{else.} \end{cases} \tag{4}$$

4.2. Profit-to-utilization ratio model

It is the ratio of actual resource usage costs to overall prices, including VM overhead startup, idle time, and shutdown. Higher the PUR level, higher the provider's throughput, diminish the energy expense, and superior reliability of the processing server. Mathematically, its formula is given based on Eq. (5).

$$PUR = 1 - \left(\sum_{j=1}^{m} \frac{\tau_i^s + in_j + \tau_i^e}{\mathcal{P}_{C_j} \times [\tau_i^s + in_j + \tau_i^e]} \right) \tag{5}$$

In the subsequent sections, we will discuss the key components necessary for enhancing the PUR value.

a. **Power Model:** This model depends mainly on a CPU, storage disk, network interface, and other operating circuits. CPUs dominate energy consumption compared to other computational resources. We made changes to the DVFS process which allow for the adjustment of the CPU clock frequency and supply voltage, resulting in significant energy savings. The power consumption can be estimated using the following formula [23].

$$\mathcal{P}_j = \mathcal{P}_S + h\left(\mathcal{P}_C + \mathcal{P}_d\right) = \mathcal{P}_S + h\left(\mathcal{P}_C + C_\epsilon f^\alpha\right) \tag{6}$$

Let $\mathcal{P}_S$ represent the power dissipation attributed to static factors, $\mathcal{P}_C$ denote the active power independent of frequency, and $\mathcal{P}_d$ designate the dynamic power dependent on frequency. $\mathcal{P}_d$ is responsible for the majority of energy consumption and can be estimated using Eq. (7).

$$\mathcal{P}_d = C_\epsilon \times \mathcal{V}_{dd}^2 \times f \tag{7}$$

where C_ϵ represents the effective loading capacitance, $\mathcal{V}_{dd}^2$ denotes voltage's values. f is the clock frequency. Based on [24], the clock of the CPU frequency directly affects the supply voltage, $f \propto \mathcal{V}^\beta$ where $(0 < \beta < 1)$. When the frequency is reduced, the supply voltage also decreases in a linear manner. We assume that the cost of power is dependent on the frequency, $\mathcal{P}_d \propto f^\alpha$ [25], where $\alpha = 1 + 2/\beta \geq 3$. The expense can be represented by a time-based formula, as calculated by Eq. (8).

$$EC_j^f(\mathcal{W}^i) = \mathcal{P}_{C_j} \times \frac{\gamma_j}{f_j} + C_\epsilon \times \gamma_j \times f_j^2 \tag{8}$$

where γ is the computational cost of computing any task using frequency f_j. The total energy consumption of computing batch of workflows in the task-graph is estimated by Eq. (9).

$$E_{total} = \sum_{j=1}^{n} EC_j^f(\mathcal{W}^i) \tag{9}$$

b. **Fault-Tolerant Model:** Placement reliability pertains to the probability of an error-free rate when executing workflow applications on the underlying cloud servers [26]. Many factors cause application failures, such as power failures, bandwidth shortages, excessive memory, etc. [27]. Due to these failures, cloud servers and communication links are susceptible to users' inability to use them. As a result, fault tolerance in real-time applications is becoming a common challenge for distributed computing systems. This study employs the Poisson distribution to evaluate the dependability of task execution on VMs. To specify the reliability issue, we establish the following hypotheses:

- The mega mapper broker waits for a specific time, defined as δ, for a task processing completion.
- The mega mapper broker is responsible for detecting failures in the network.
- Failure of different resources is statistically independent.
- The probability $\mathcal{R}(\mathcal{S}_i, \Delta\tau)$ represents the likelihood that the server $\mathcal{S}_i$ remains operational without any failures over a duration of $\Delta\tau$.
- The probability $\mathcal{R}(\mathcal{L}_{i,j}^\tau, \Delta\tau)$ denotes the likelihood that the link $\mathcal{L}_{i,j}$ remains operational without any failures while task τ is being transferred over a duration of $\Delta\tau$.

The reliability of the server $\mathcal{S}_i$ during time interval $\Delta\tau$ has an exponential probability density function, derived as $\mathcal{R}(\mathcal{S}_i, \Delta\tau) = e^{-F_i\Delta\tau}$. Given this derivation and based on the third hypothesis, the reliability of the assigned tasks at the application level depends entirely on the reliability of all servers on the cloud platform, Eq. (10) [28].

$$\mathcal{R}(\mathcal{L}_{i,j}^\tau, \Delta\tau) = \prod_{i=1}^{\alpha} \mathcal{R}(\mathcal{S}_i, \Delta\tau) = e^{-\left(\sum_{i=1}^{\alpha} F_i\right)\Delta\tau} \tag{10}$$

The initial hypothesis affirms that the Mega Mapper will examine the priority status of task τ_i. If it does not receive the processing results after δ time, this means there is a failure rate. To improve the reliability of our application placement, we integrated *the reliability enhanced time function* model proposed in [28] to compute the task τ_i on the processing server $\mathcal{S}_i$ and the transfer data of size η on the bandwidth $\mathcal{B}_{i,j}$ with the delay link from the initiated node to the reached node $\mathcal{D}_{i,j}$, as shown in Eqs. (11) and (13), respectively.

$$\mathcal{R}E(\tau) = \frac{\xi_i}{1 - F_j} \tag{11}$$

$$\xi_i = \frac{\zeta_i}{\mathcal{P}_j} \tag{12}$$

$$\mathcal{R}E(\eta) = \frac{(\frac{\eta}{\mathcal{B}_{i,j}}) + \mathcal{D}_{i,j}}{1 - F_{i,j}} \tag{13}$$

4.3. Multi-dimensional resource partition model

The servers have d-dimensional resources. When a certain dimension resource is exhausted, the servers will be fully loaded, and they cannot host further applications. Therefore, a new server starts up to host them, which might result in resource fragments

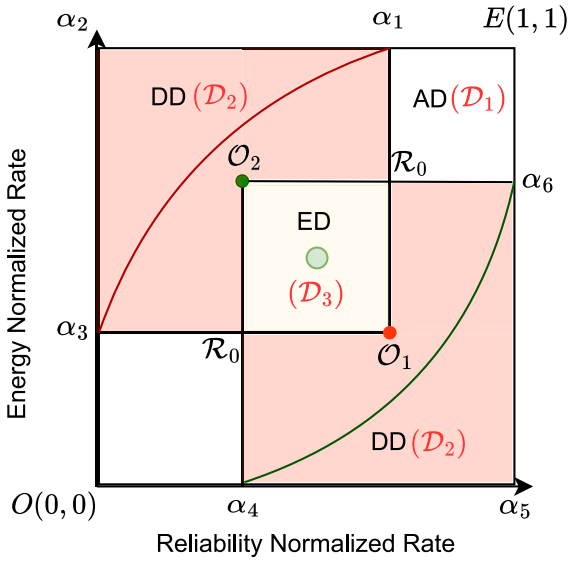


Fig. 2. Multi-dimensional resource partition model [29].

(i.e., resource leakage). This paper focuses mainly on reducing resource fragmentation by balancing resource utilization in all dimensions.

Building upon the ideas presented in [29,30], we incorporated a multi-dimensional resource partition instance to construct a d-dimensional space partition model for the servers, which allows us to choose the optimal resource utilization for mapping VMs. Within this model, we initially define the current usage rate of server S_j as follows.

Definition 2 (*Utilization Rate*)**.** For a server S_j, the dth dimensional resource utilization ratio is denoted as $\mathcal{Y}_S$, where $1 \leq d \leq D$. The utilization rate of server S_j can be represented as $UR(S_j) = (U_S^1, U_S^2, \ldots, U_S^D)$, with each utilization value corresponding to a point in the multi-dimensional model.

To simplify the multi-dimensional model paradigm, we consider Fig. 2 as an example according to [29] when $D = 2$. As shown in the figure, we have two coordinates. The first coordinate is $\mathcal{O} = (0, 0)$, which indicates the initial point when the server S_j is idle. In contrast, the second coordinate is $E = (1, 1)$. This means that all of the dimensional resources have been used up. The unit square has been divided into three-quarter circles [29,30].

- *Acceptable Domain (AD)*: This domain determines that d-dimensions of processing servers are appropriately balanced and that there are few resource fragments in the server S_j. This is an acceptable domain for all operating servers.
- *Equilibrium Domain (ED)*: This domain signifies that the utilization of d-dimensional resources does not indicate any obvious disbalance and has a relative balance overall.
- *Disequilibrium Domain (DD)*: This domain specifies an apparent disequilibrium of d-dimensional resource utilization, and servers may have overextended resource fragments. In an efficient application placement scheme, this phenomenon should be avoided.

The part of the acceptance area D_1 is a quarter circle with a central E and r_0 radius, where $r_0 \in [0, 1]$. The area of imbalance is partitioned into two equal parts, each comprising a quarter circle with radius r_0 such that $r_0 \in [0, 1]$. The midpoint of the red quarter circle is represented as $\mathcal{O}_1$, whereas the green quarter circle is centered at $\mathcal{O}_2$. The equilibrium area is between the square's edges and the three-quarter circle. In general, the determination function f for the d-dimensional resource usage model is represented as [30].

$$f_S = \begin{cases} AD, & \text{if } S_j \text{ belongs to the domain } D_1; \\ ED, & \text{if } S_j \text{ belongs to the domain } D_3; \\ DD, & otherwise. \end{cases} \tag{14}$$

5. Algorithm design

The reliability of the application placement is crucial and even more critical than the exact schedule length. Optimum reliability of application placement must be achieved by reducing cloud servers' energy consumption. Application placement in clusters consists of two crucial stages, namely application priority calculation and application-processor pairing.

This section details three algorithms designed to optimize the dependability of task execution and reduce energy consumption in parallel applications running on heterogeneous cloud resources. The first algorithm, namely the Energy-aware Dynamic Voltage and Frequency Scaling (EDVFS) algorithm, calculates an optimal or almost optimal frequency, reducing energy consumption at the lowest final time in a heterogeneous processor. The second algorithm proposed in this work is the Reliability-aware Heterogeneous Fault-Tolerant (RHFT) algorithm. Compared to EDVFS algorithms, RHFT algorithms focus mainly on avoiding high failure rates of servers and communications links. In such a heuristic, the distinctive feature is forming an active high-performance computing server to host assigned tasks. In addition, a new algorithm, namely the Energy-aware stepwise Reliability Maximization (ERMax) algorithm, selects the best combination of task-server while ensuring energy reduction and reliability optimization.

5.1. Impact of task execution on energy and reliability

This section focuses on schedule components that allow higher reliability and low energy costs to be achieved without increasing the task execution time during application placement. Processor boundedness, ϑ_p, is defined as a ratio between 0 and 1 according to [31]. When $\vartheta_p = 1$, the performance of a specific job ultimately depends on the allocated server frequency (i.e., the processor-bounded), and the execution time is in reverse proportion to the processor frequency. However, power consumption does not change monotonically with frequency increases due to frequency-independent active power [25]. This deduces that low frequencies may not always reduce energy costs, and when tasks are computed to achieve the lowest energy costs, the optimal voltage and frequency pairs are needed [25]. Increased frequency levels will increase the reliability of application placement simultaneously. This implies that achieving both high reliability and low energy costs is mutually exclusive. and that an effective balance approach is needed to improve the throughput of application placement.

5.2. Primitive phases

The three algorithms described below consist mainly of two phases. The following part will briefly discuss them as explained below.

a. **Critical-Path-Of-Tasks (CPOT):** At this stage, the widely-used Longest Path (LP) algorithm is utilized to determine the longest computing path (i.e., critical path) in the application task graph (i.e., application layer). The critical path comprises of tasks with EST matches the LST. This helps to determine how long it will take to complete each task.
b. **Task assigned priority:** In order to meet task placement requirements, a ranking estimate model is proposed to generate task placement sequences while task dependency is satisfied.

$$\mathcal{RK}_i = \overline{\mathcal{R}\tau_i} + \max_{\tau_j \in suc(\tau_i)} \left(\mathrm{T}\tau_{i,j} + \mathcal{RK}_j\right) \tag{15}$$

The set $suc(\tau_i)$ represents the immediate neighbors of task τ_i. The running time of task τ_i is denoted as $\mathcal{R}\tau_i$, and $\mathrm{T}\tau_{i,j}$ represents the transfer time between source and destination servers. The ranking is determined by recursively moving from the lower levels of the task graph to the higher levels.

5.3. The key steps proposed by the algorithm

5.3.1. The Energy-aware Dynamic Voltage and Frequency Scaling (EDVFS) algorithm

The algorithm incorporates the DVFS techniques to lower energy usage. The main goal is to find the best frequency for the computing server to carry out the assigned tasks without prolonging the schedule's duration or compromising its computational complexity.

Fig. 3 shows the existence of the local minima of energy consumption at different frequencies. The diagram shows that the optimal frequency is derived from the following [28], when $\delta = 105$, $\alpha = 6.5$, and $\beta_{cpu} = 1$, respectively.

$$\mathcal{S}_j^f(\mathcal{OPT}) = \sqrt[3]{\frac{\delta_j}{2\alpha_j}} \tag{16}$$

As stated earlier, the ideal CPU frequency is directly influenced by static variables in a monotonic manner. It is possible to calculate the local minima in advance. $\mathcal{S}_j^f(\mathcal{OPT})$ is not bounded to $[\max\left(\mathcal{S}_j^f(min), \mathcal{S}_j^f(\tau_i)\right), \mathcal{S}_j^f(max)]$, which is the frequency range for server $\mathcal{S}_j$ to utilize to satisfy the task's deadline constraint.

As the existing retail processors have limitations in their frequency support, we will select a frequency from the designated range $[\max\left(\mathcal{S}_j^f(min), \mathcal{S}_j^f(\tau_i)\right), \mathcal{S}_j^f(max)]$ according to [28]. Considering Fig. 3 as an example, the optimal frequency is 2.0 GHz, which is the nearest to the frequency $\mathcal{S}_j^f(\tau_i)$.

The lower the CPU frequency, the longer the task execution time is. This suggests that the values of these two measurements should be optimized at an improved service delivery time ratio for application placement. To do this, we first calculate the performance time for each task on the computing server in accordance with Eq. (17).

$$\mathcal{PR}\tau_j^i = \mathcal{C}\tau\left[\mathcal{S}_j^f(\tau_i)\right] \times \frac{\mathcal{W}_d}{e\mathcal{C}\tau} \tag{17}$$

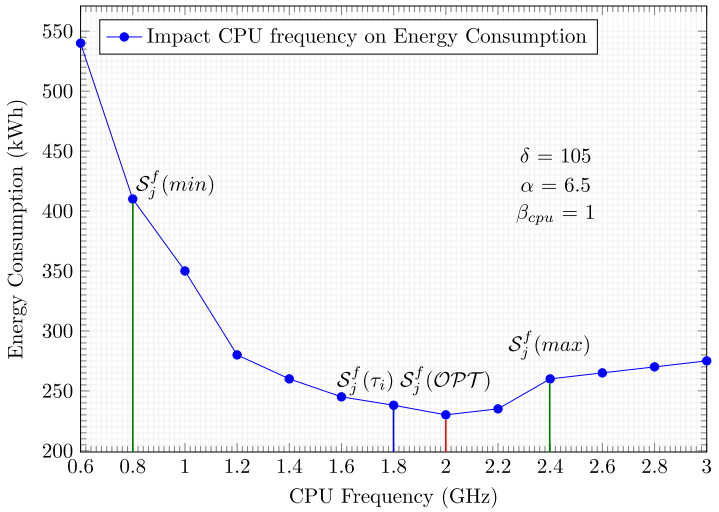


Fig. 3. Energy consumption vs. CPU frequency (Execution time = 1 h).

where $\mathcal{W}_d$ is the workflow's deadline, $eC\tau$ is the earliest completion time of workflow, and $C\tau\left[S_j^f(\tau_i)\right]$ is the computing time of task τ_i on server S_j utilizing frequency f, which is estimated according to Eq. (18).

$$C\tau\left[S_j^f(\tau_i)\right] = C\tau\left[S_j^f(max)\right] \times \left(\vartheta_p\left(\frac{C\tau\left[S_j^f(max)\right]}{C\tau\left[S_j^f(\tau_i)\right]} - 1\right) + 1\right) \tag{18}$$

Therefore, we combine Eqs. (17) and (18) to estimate the minimum acceptable frequency according to Eq. (19)

$$S_j^f(\tau_i) = \frac{\vartheta_p \times S_j^f(max) \times C\tau\left[S_j^f(max)\right]}{\vartheta_p \times S_j^f(max) + \mathcal{PR}\tau_j^i - C\tau\left[S_j^f(max)\right]} \tag{19}$$

Algorithm 1: The Energy-aware Dynamic Voltage and Frequency Scaling (EDVFS) Algorithm

Input: *Servers each initialized to its maximum instantaneous power using the DVS=*$\{dvs_1, dvs_2, ..., dvs_\tau\}$.
Output: *A mapping scheme that consumes minimum power without violating minimum possible makespan.*

Compute Critical Path (CP) for incoming tasks of a workflow using well-known polynomial time (LP) algorithm.
Compute $\mathcal{RK}_i$ for each task, τ_i where $\tau_i \in T$, via spanning the task graph from the end server according to Eq. (15).
Compute the optimal frequency for each server in the network graph according to Eq. (19).
while (*the priority queue,* $\mathcal{PQ}(\mathcal{RK}_i)$*, is not empty*) **do**
  for $\forall \tau_i \in \mathcal{PQ}(\mathcal{RK}_i)$ **do**
    for $\forall S_j \in \mathcal{DVS}$ **do**
      Compute energy cost of server S_j on frequency f to execute task τ_i, $E\left(S_j^f(\tau_i)\right)$, according to Eq. (8).
      Sort th servers in decreasing order of their current energy consumption, $\mathcal{SQ}$: $E_1 \geq E_2 \geq ... \geq E_m$.
    end
  end
  for $\forall S_j \in \mathcal{SQ}$ **do**
    if *the summation of energy consumption of server* S_j *satisfies the minimum possible makespan* **then**
      $S_p \leftarrow S_j^f(\tau_i)$
    end
    else
      Skip S_j and delete the server from *SQqueue*
    end
  end
  Compute the failure rate of server S_p by calling the RHFT algorithm.
end

The EDVFS algorithm (Algorithm 1) is a placement algorithm for applications with a limited number of heterogeneous processing servers. The procedure comprises two main steps: first, the priority calculation step estimates the critical path of tasks within the

workflow and generates a suitable topological ordering of these tasks. Then, the server selection phase follows, in which the available task is assigned to the most energy-efficient server. The primary focus of Steps 1 and 2 in Algorithm 1 is to calculate the critical path and determine the sequence for placing each task. Step 3 involves estimating the optimal frequency for each server. Steps 4–20 form an outer loop within the EDVFS algorithm, ensuring the effective placement of assigned tasks on suitable servers. In Step 7, the energy cost of each processing server is computed according to Eq. (8). The energy-optimized queue is constructed in Step 8. Steps 11–18 guarantee that the selected servers do not degrade the tasks' makespan. After traversing all the servers in Steps 11–15, the servers that fail to satisfy the makespan constraint will be eliminated from the energy-optimized queue. Step 10 calculates the failure rate of the candidate server by conducting the RHFT algorithm.

5.3.2. The Reliability-aware Heterogeneous Fault-Tolerant (RHFT) algorithm

The RHFT algorithm is another essential algorithm that employs fault-tolerance techniques to monitor the status of assigned tasks. The algorithm begins the verification process after waiting for a period equivalent to the task's execution time, which is referred to as delta. In the event that the task execution does not yield the desired results, the algorithm identifies the cause of the failure. Furthermore, if the server crashes, the algorithm scrutinizes the crash information to locate a compatible VM to host the task and reinstates the delta period.

The primary objective of this algorithm is to enhance the reliability of computational tasks. To achieve this objective, we introduce a placement metric denoted as $\mathcal{P}m$ derived from the approach described in [26]. The purpose of this metric is to identify the optimal task placement strategy.

$$\mathcal{P}m_j^i = \begin{cases} \alpha \times \dfrac{EC\left(S_j^f(\tau_i)\right) - EC\left(S_j^f(min)\right)}{EC\left(S_j^f(max)\right) - EC\left(S_j^f(min)\right)} + \\ \beta \times \dfrac{\mathcal{R}\left(S_j^f(max)\right) - \mathcal{R}\left(S_j^f(\tau_i)\right)}{\mathcal{R}\left(S_j^f(max)\right) - \mathcal{R}\left(S_j^f(min)\right)}, \\ \alpha + \beta = 1. \end{cases} \tag{20}$$

Here, $EC\left(S_j^f(\tau_i)\right)$ denotes the energy consumed by server S_j when running task τ_i at its optimal frequency. $EC\left(S_j^f(min)\right)$ and $EC\left(S_j^f(max)\right)$ represent the energy consumed by server S_j when operating at its minimum and maximum frequencies, respectively. On the other hand, $\mathcal{R}\left(S_j^f(\tau_i)\right)$ refers to the reliability of executing task τ_i on server S_j at its optimal frequency. Similarly, $\mathcal{R}\left(S_j^f(min)\right)$ and $\mathcal{R}\left(S_j^f(max)\right)$ indicate the reliability of executing task τ_i on server S_j at its minimum and maximum frequencies, respectively. The parameters α and β represent the weightage assigned to the energy and reliability metrics, respectively. Depending on which metric is prioritized, the corresponding parameter (α or β) is assigned a higher value.

As demonstrated in Algorithm 2, Step 1 involves creating a computing environment that comprises processing servers and communication links, ensuring that resource homogeneity and connectivity completeness are maintained. This step takes into account the initial computation and transportation time costs. Step 2, 3, and 15 involve creating a level queue to segregate workflow tasks, a task queue to sort the tasks based on their priorities, and a server queue to rank the servers in descending order based on their energy consumption, respectively. Step 4 utilizes a widely recognized polynomial algorithm to compute the CP for the tasks in the assigned workflows. Steps 5–14 guarantee that the tasks along the CP are prioritized and scheduled on high-performance processing servers. Following the traversal of all servers in Steps 16–30, the most appropriate servers that can provide the best performance-to-reliability ratio are selected to host the mapped tasks. In Step 19, the reliability function of each server is estimated using the function described in Section 4.2.

5.3.3. The Energy-aware stepwise Reliability Maximization (ERMax) algorithm

To leverage the profit-to-utilization ratio, the algorithm uses energy and reliability to trade-off the service delivery time ratio for application placement to enhance the level of efficiency and reliability. It only allocates non-critical applications to processing servers, thereby improving reliability rates. The process ensures that the most important task, which is part of the CP, is kept on the same server. Additionally, it assigns a specific level of reliability and energy cost to ensure the computation is carried out successfully.

The ERMax algorithm takes into account two conditions. The first condition involves only remapping tasks that have a new finish time that does not exceed the latest finish time (LFT). The second condition involves selecting a task that is ready to start at the latest starting time (LST). Moreover, the algorithm picks a server that is directly connected to the server on which the predecessor task is scheduled to run.

Algorithm 3 outlines the following steps. Firstly, in Step 1, the task queue, TQ, is created and used to sort workflow applications in decreasing order of their priorities, using the topological sort algorithm. Next, in Step 2, the critical path for tasks within the workflow is estimated using a well-known polynomial-time algorithm. To ensure improved reliability during the execution of tasks that are not part of the critical path, Steps 3–13 are implemented. In Step 6, a new task queue, denoted as nCP, is created, which

Algorithm 2: The Reliability-aware Heterogeneous Fault-Tolerant (RHFT) Algorithm

Input: *Energy-optimized a bounded of servers, each initialized with its optimum frequency.*
Output: *A mapping scheme that retain higher reliability without violating minimum possible makespan.*

1 In the context of a computing environment that is heterogeneous and prone to faults, denoted by $\mathcal{G}\tau^{\star}$, each server and communication link are linked to a fixed rate of failure.
2 Create a level queue, $\mathcal{LQ}$, to separate the workflow tasks into different levels.
3 Create a task queue, TQ, to sort the DAG-structured workflow tasks in decreasing order of their priorities by conducting topological sorting algorithm.
4 Compute the CP for requests in a workflow using well-known polynomial time (LP) algorithm.
5 **while** (*the level queue, $\mathcal{LQ}$, is not empty*) **do**
6 **for** *each* $\tau_i \in TQ$ **do**
7 **if** $\tau_i \in \mathcal{CP}$ **then**
8 $\mathcal{C}\tau \leftarrow \tau_i$
9 **end**
10 **else**
11 skip τ_i
12 **end**
13 **end**
14 **end**
15 Using the topological sorting approach, generate a server queue, denoted as $\mathcal{SQ}$, to arrange the servers in a descending order based on their energy consumption.
16 **while** (*the server queue, $\mathcal{SQ}$, is not empty*) **do**
17 **for** $\forall \mathcal{S}_j \in \mathcal{SQ}$ **do**
18 **for** $\forall \tau_i \in \mathcal{C}\tau$ **do**
19 Compute the reliability of the task τ_i on the server $\mathcal{S}_j$ according to the reliability function given in Section 4.2.
20 **if** $\mathcal{R}_j^i < \mathcal{R}_{j-1}^i$ **then**
21 Skip $\mathcal{S}_j$
22 **end**
23 **else**
24 Assign the marked task τ_i to the server $\mathcal{S}_j$
25 **end**
26 **end**
27 Compute a new reliability function according to the reliability function given in Section 4.2.
28 Update the server queue, $\mathcal{SQ}$.
29 **end**
30 **end**

consists of tasks with the minimum LST among those in the TQ queue. Subsequently, in Steps 14–17, a new finish time is calculated for every job in the $nC\mathcal{P}$ queue, according to the equation mentioned in Eq. (21).

$$\mathcal{N}_{F\tau}^{i} = \mathcal{SL}_j + \frac{\mathcal{C}\tau\left[S_j^f(\tau_i)\right]}{1 - F(S_j)} + \frac{\frac{s^i_{pre(\tau_i)}}{\mathcal{B}_{upst_{i,j}}}}{1 - F_{upst_{i,j}}} \tag{21}$$

In the given equation, the term $\mathcal{SL}(\mathcal{S}j)$ represents the duration of the schedule for server $\mathcal{S}j$, and $s_{pre(\tau_i)}$ denotes the data size of the predecessor tasks of τ_i, $\mathcal{B}_{upst_{i,j}}$ denotes the bandwidth between the two upstreaming servers, $1 - F(\mathcal{S}_j)$ indicates the failure rate of server $\mathcal{S}j$, and $1 - F_{upst_{i,j}}$ is the failure rate of the link connecting servers $\mathcal{S}_i$ and $\mathcal{S}_j$.

In Step 15, the minimum reliability costs for the application are set to an extremely high value of infinity, and at the outset, these applications are not marked, meaning they have not yet been remapped. Steps 19–37 ensure that servers with direct connections to the servers on which the predecessor tasks of τ_i are scheduled are chosen. The execution reliability of each task on servers is estimated in Step 23. Further, Steps 24–29 guarantee the most reliable service for each task. A new reliability value is computed in Step 27.

5.4. The process of combining the EDVFS, RHFT, and ERMax algorithms

In this subsection, the sequential operation and interconnection of the three algorithms are described. Initially, the workflow applications are categorized into the critical and non-critical paths. Applications belonging to the critical path are given higher priority and scheduled first on the cloud servers. Once the task modules on the critical path are ready for processing, the initial algorithm, EDVFS, commences its execution in the following manner.

Algorithm 3: The Energy-aware stepwise Reliability Maximization (ERMax) Algorithm

Input: *Application graph: DAG-structured workflow, Network graph: bounded energy optimized and high reliable servers.*
Output: *Application placement scheme with reliability improvement.*

Create a task queue, TQ, to sort the DAG-structured workflow tasks in decreasing order of their priorities by conducting topological sorting algorithm.
Compute the CP for demands in a workflow using well-known polynomial time (LP) algorithm.
while (*the task queue, TQ, is not empty*) **do**
  for *each* $\tau_i \in TQ$ **do**
    if $\tau_i \notin \mathcal{CP}$ **then**
      Form a new task queue, denoted as $n\mathcal{CP}$, to arrange the tasks that do not belong to the critical path, based on their minimum latest start time, represented as $\mathcal{LST}$.
      $n\mathcal{CP} \leftarrow \tau_i$
    end
    else
      skip τ_i
    end
  end
end
for *each* $\tau_i \in n\mathcal{CP}$ **do**
  Set the minimum reliability cost, represented as $min\mathcal{RC}$, to be an extremely large value such as infinity, denoted by ∞.
  Calculate a fresh finish time, referred to as FT, for the task τ_i that is assigned to the server S_j, using the equation stated in Eq. (21).
end
Create a server queue, SQ, to sort the servers in decreasing order of their energy consumption by conducting topological sorting algorithm.
while (*the server queue, SQ, is not empty*) **do**
  for $\forall S_j \in SQ$ **do**
    if S_j *has the* τ_i*'s predecessor tasks* **then**
      Mark S_j.
      Compute the reliability of the task τ_i on the server S_j according to the reliability function given in Section 4.2.
      if $\mathcal{R}_j^i < \mathcal{R}_{j-1}^i$ **then**
        Skip S_j
      end
      else
        Allocate the task τ_i, which has been marked or identified, to the server denoted as S_j.
      end
    end
    else
      Skip S_j
    end
    Compute a new reliability function according to the reliability function given in Section 4.2.
    Update the server queue, SQ.
  end
end

The EDVFS algorithm takes the applications as input and assesses the frequencies of each server. Each server maintains a set of frequencies, and the objective is to determine the optimal or near-optimal frequency. This computed frequency should result in the highest performance, achieving a balance where both server energy consumption and computational complexity are optimized. Once the optimal frequency is calculated using Algorithm 1, the servers along with their respective optimal frequencies are inputted into Algorithm 2.

The RHFT algorithm specifically chooses servers from the queue that have the capability to execute applications with the highest reliability rates. To achieve this, the algorithm calculates the reliability cost model for running each application on a particular server using the optimal frequency. During this stage, servers that fail to meet the objective of achieving a reliability ratio are excluded from the queue. Subsequently, the workflow applications are dispatched to servers that are both reliable and aware of the optimal frequency requirements.

The ERMax algorithm is an extension of the RHFT algorithm specifically designed for non-critical path applications. It takes as input the optimized servers, which are servers with awareness of both frequency and reliability obtained from Algorithm 2. The algorithm organizes the non-critical path applications based on their minimum latest start time. Additionally, it utilizes the dependencies between applications to further enhance the reliability cost model. It considers servers with allocated tasks, assigned by Algorithm 2, as their reliability ratios are calculated by Algorithm 2. In order to assign a non-critical application to one of these

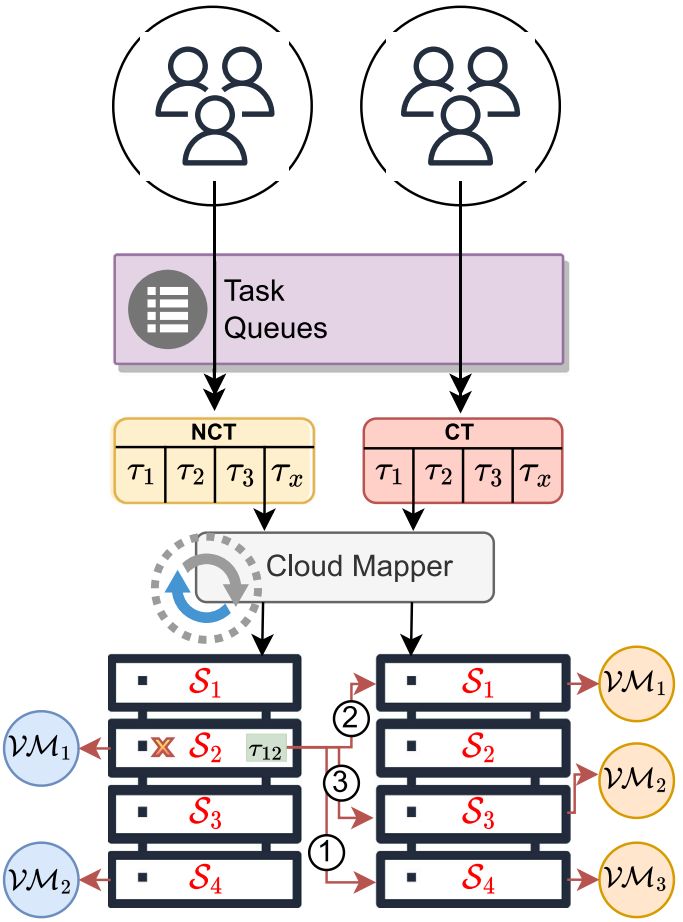


Fig. 4. Example of solid fault tolerance.

servers, Algorithm 3 examines the predecessor tasks (i.e., critical tasks already scheduled by Algorithm 2) of the new non-critical task. If the new task has predecessor tasks assigned to a specific server, Algorithm 3 assigns the new task to the same server.

5.5. An illustrative example analysis

Fig. 4 illustrates an instance of strong fault tolerance. Our assumption is that there are two clusters located at different sites. Each cluster consists of four computing devices, each with its distinct numeral of VMs and capabilities. A group of job tasks is allocated and scheduled on VMs 1, 2, 3, 4, and 5 across servers 1, 2, 3, and 4 in clusters 1 and 2, respectively. To enhance reliability, the system regularly undergoes testing to evaluate the likelihood of failures. This testing occurs after a specific time period, which is equal to the duration needed to execute a task (denoted as δ). However, each task has a finite completion time. We assume that the task execution process operates smoothly until $\tau = 12$, when a particular task on VM-1 fails at the expected time due to the crash of server 2 in the first cluster. At this stage, we consider alternative solutions to address the issue.

- The algorithm initially retrieves information about the task running on the virtual machine (VM-1) of the crashed server. It then attempts to locate the most suitable running VM within the same cluster and returns to the initial stage. This solution does not require the initiation of a new VM or the transfer of tasks to neighboring clusters, thereby avoiding any additional communication overhead.
- When the current active server in the first cluster does not have enough resources to accommodate tasks, the algorithm retrieves information about the active server in the second cluster. It then assigns tasks to that server, which incurs additional communication costs.

6. Performance analysis and evaluation

6.1. Evaluation criteria measurement

The Bi-OWSP paradigm has compared with other algorithm with respect to the following five criteria:

- Probability of Failure (PoF): The PoF is regarded as one of the initial performance metrics for our comparison. The algorithm assigns a PoF value to each task based on the optimal frequency allocated. The definition of PoF is as follows.

$$PoF = 1 - \prod_{i=1}^{\alpha} \mathcal{R}\left(S_j^f(\tau_i)\right) \tag{22}$$

- Energy consumption Ratio (ECR): The energy Consumption ratio is another significant performance criterion for a given task that is defined as follows.

$$ECR = \frac{E_{total}}{\sum_{\forall \tau \in CP} \min_{\forall pre_i \in S_j} \left\{ \mathcal{P}_{C_j} \times \frac{\gamma_j}{f_j} + C_e \times \gamma_j \times f_j^2 \right\}} \tag{23}$$

- Profit-to-Utilization Ratio (PUR): The PUR criteria capture the significance of balancing energy reliability across applications by optimizing their placement. The PUR value, expressed as a ratio, represents the operational cost of the server in relation to the total expenses, encompassing additional costs associated with VMs such as their activated, inactivated, and close-down time. A higher PUR value indicates enhanced system throughput and maximized profit for the provider. The mathematical estimation of PUR is as follows.

$$PUR = 1 - \left(\sum_{j=1}^{m} \frac{\tau_i^s + in_j + \tau_i^e}{\mathcal{P}_{C_j} \times [\tau_i^s + in_j + \tau_i^e]} \right) \tag{24}$$

- Guarantee Ratio (GR): The guarantee ratio represents the proportion of applications that are completed successfully within the specified deadlines, relative to all the applications that have been assigned. A higher guarantee ratio indicates a higher level of reliability in execution. The procedure for calculating the guarantee ratio is described as follows:

$$GR = 100\% \times \frac{\text{ready-to-execute task}}{\text{total tasks in the task-graph}} \tag{25}$$

- Energy-Reliability Trade-off (ERT): To demonstrate the superior tradeoff between energy and reliability achieved by our algorithm in comparison to others, we computed a third parameter called ERT (Energy-Reliability Tradeoff) for the entire scenario. This was undertaken to highlight the connection between energy usage and system reliability and to showcase our algorithm's ability to strike a favorable balance between the two.

$$ERT(\alpha_j) = \frac{\min_{\alpha_i \in alg}[EC(\alpha_i)]}{EC(\alpha_j)} \times \frac{\min_{\alpha_i \in alg}[\mathcal{RC}(\alpha_i)]}{\mathcal{RC}(\alpha_j)} \tag{26}$$

where $alg = \{\alpha_1, \alpha_2, \ldots, \alpha_i\}$ represent a collection of algorithms. The highest attainable ERT value for any algorithm is 1, signifying optimal performance. This can only be achieved when the algorithm consistently outperforms others paradigms regarding energy usage and system reliability, surpassing their performance.

6.2. Simulation characteristics

To demonstrate the effectiveness of the Bi-OWSP, we carried out a comprehensive set of simulation experiments using the CloudSim Plus simulator [32,33]. This simulator is designed to analyze and simulate research concepts specifically for data centers and is an enhanced edition of the original software [34]. For further information regarding the simulation parameters, please refer to Table 3.

The experiments included a set of 100 cloud servers, each with processing capacities ranging from 500 to 1500 MIPS and network bandwidths varying from 10 to 1024 Mbps [35]. The failure rate for each server was adjusted between 10^{-5} and 10^{-7} [15], and was 0.0005 to 0.000005 for the communication link varied [36,37]. Virtual machines were dynamically allocated or unallocated based on the job's requirements. The VM initiating time was 100 s, and the terminating time was 8 s [38].

The investigation took into account approximately 650 to 1250 application modules. The sizes of input and output packets ranged from 0.18 to 0.25 MB. The required duration for the service was specified between 0.500 and 0.700 s. The volumes of instructions varied from 20 to 30 [39].

6.3. Comparable algorithms

To assess the effectiveness of the proposed Bi-OWS algorithm, we conducted a quantitative comparison of its performance against five baseline algorithms: FDRGS [14], SERAS [16], ReadyFS [19], dual-algorithm [20], and EQ-VMC [21].

Table 3
Features of the simulation.

Characteristics	Quantitative specification
Simulation settings	
Length of simulation time	700 s
Duration between consecutive rounds of placement	0.065–0.193 s
Total quantity of processing units	100 nodes
Frequency of request arrival	15–65 incoming requests/s
Workflow requirements	
Number of assigned demands	100–800 tasks
Maximum time allotted for service delivery	0.500–0.700 s
Input packet size	0.18–0.25 MB
Output packet size	0.18–0.25 MB
Number of instructions	20–30 TI
Memory required	50–250 MB
User's budget	0.80–1.00 $ per application
Computing nodes	
Quantity of cloud units	100 computing appliances
Price list for services	0.025–0.06 $/s
Computation expense	0.4–0.8 $/s
Expense of networking	0.002–0.007 $/s
The frequency of link failures	0.0005–0.000005
Computing power	500–1500 MIPS
VM weight	2.5 GB
VM activated time	100 s
VM closed time	8 s
The rate of server failures	10^{-5}–10^{-7}
Communication channel capacity	10–1024 Mbps

7. Experimental results

The performance parameters of the Bi-OWSP were assessed in two separate settings. The initial setting involved a synthetic workflow comprising tasks ranging from 100 to 8500. The evaluation utilized the Pegasus workflow management systems [40,41] in the second setting and focused on optimizing the main metrics influencing the provider's throughput. The structure of the synthesized workflow used to assess the proposed algorithms is depicted in Fig. 5.

The workflow depicted in Fig. 5 is artificial and adheres to a directed acyclic graph. This graph comprises a series of tasks starting from S and culminating in task E. Each of the edges, denoted as $e_{i,j}$, connects sequential tasks τ_i and τ_j. Associated with each edge $e_{i,j}$, there exists a weight denoted as $\mathcal{W}(i,j)$ or $\xi_{i,j}$, indicating the size of the input data transferred from the source task τ_i to the destination task τ_j. In essence, these weights signify the computing dependency between the tasks τ_i and τ_j.

Our workflow is a meticulously organized process where input data is passed from one task to the next. Each task performs its routine procedure, and its complexity is determined by its associated cost function. However, in real-world scenarios, a job's complexity relies on both the computational intricacy of its operation and the specific execution details of the algorithm it employs. Our model emphasizes that none of the tasks can commence execution until they have received all the necessary data from their preceding tasks. In cases where a job has multiple access or leave points, we implement a zero-complexity application that connects to all triggering and terminating tasks in the workflow.

7.1. Synthesized scientific workflow

Fig. 6(a) illustrates a comparison of four algorithms using different workload sizes, based on the PoF metric. These algorithms include Bi-OWSP, ReadFS, FDRGS, and SERAS. The algorithms Bi-OWSP and ReadFS demonstrate the best results, indicated by the lowest failure rate. This is because they take into account resource failures, such as server and communication link failures, when scheduling and executing tasks on virtual machines (VMs). The slight variation between these two algorithms stems from the fact that Bi-OWSP excludes servers with failure probabilities exceeding a specified limit within a defined reference period for task hosting. Conversely, the SERAS algorithm exhibits the lowest performance concerning the PoF. This is due to its disregard for Frail Failure Tolerance (FFT), similar to the FCWS algorithm, which results in a weakened performance. The FDRGS algorithm shows some improvement in performance as it considers Frail Fault Tolerance. Consequently, the PoF values of Bi-OWSP and ReadFS are more advantageous compared to those of FDRGS and SERAS, as they offer greater profitability.

Fig. 6(b) depicts a comparison of four algorithms using different workload sizes, evaluated based on the ECR metric. These algorithms are Bi-OWSP, EQ-VMC, FDRGS, and SERAS. Both Bi-OWSP and EQ-VMC algorithms employ straightforward techniques for energy optimization. As the workload sizes (representing computational complexity) increase, there is a gradual increase in ECR across the algorithms. However, Bi-OWSP and EQ-VMC demonstrate notable progress in average energy conservation, particularly when the number of applications reaches 600, 700, and 800. This highlights the significance of effective application placement

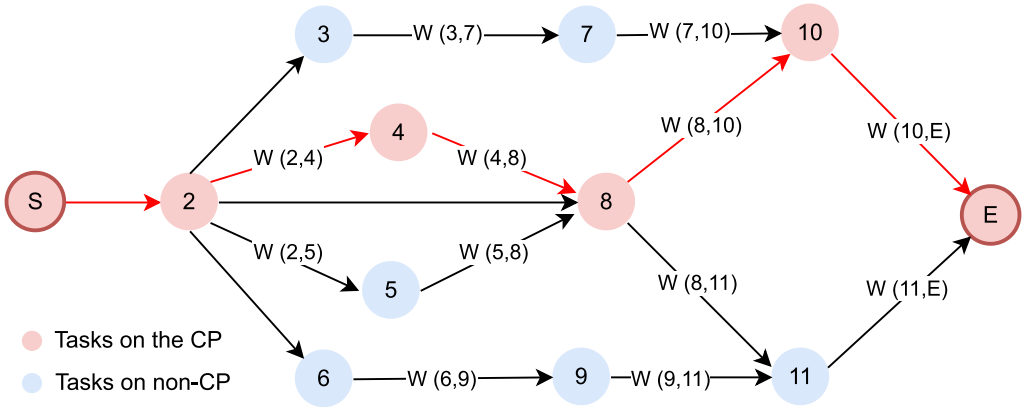


Fig. 5. The structures of synthesized scientific workflow.

in optimizing overall performance. The slight variation observed between the trends of Bi-OWSP and EQ-VMC is attributed to the former algorithm's ability to eliminate overhead by reusing allocated VMs, thereby enhancing system effectiveness. On the contrary, FDRGS and SERAS algorithms exhibit inferior performance. FDRGS, however, performs relatively better as an energy-conservative algorithm, achieving higher ECR at the cost of slightly lower reliability. In summary, based on this analysis, it can be concluded that the Bi-OWSP and EQ-VMC paradigms outperform others as the ECR increases, rendering them more practical for data-intensive applications.

The trends displayed by all algorithms in Fig. 6(c) exhibit relatively similar patterns, with Bi-OWSP and ReadFS demonstrating the best performances in terms of the PUR metric. The Bi-OWSP algorithm achieves a relatively high PUR due to its elimination of VM overhead, including startup, idle, and shutdown times, when setting up VMs in the cloud infrastructure. As a result, VMs that may result in resource leakage are rejected for hosting additional assigned tasks, leading to the selection of servers with sufficient capacity for processing. This efficient application placement ensures that tasks are assigned to capable servers, ultimately increasing the PUR. Analyzing Fig. 6(c), we can observe that ReadFS, EQ-VMC, and the dual-algorithm also yield outstanding outcomes. ReadFS considers deadline constraints and enhances resource utilization by accounting for resource failures. EQ-VMC focuses on task scheduling to decrease energy costs and increase the QoS. The dual-algorithm distribute the workload evenly and balance cloud resources to improve time allocation. The primary reason for the improvement exhibited by these four algorithms lies in their efficient management of diverse workloads from multiple data sources. They identify the optimal task-server combinations that maximize the PUR ratio.

Fig. 6(d) evaluates the performance of the same algorithms using the GR metric across different incoming sizes. Bi-OWSP, ReadFS, and the dual-algorithm exhibit superior GR values among them. The Bi-OWSP algorithm shows a slight advantage over the others due to its implementation of a multi-differential space partition mode, which optimizes resource utilization and reduces resource fragmentation. EQ-VMC and the dual-algorithm prioritize meeting deadlines, maintaining service quality, and achieving balanced processing times. The ReadFS approach strengthens its robustness by minimizing resource failures. On the other hand, the dual-algorithm benefits significantly from a balanced trade-off between load and processing time. As depicted in Fig. 6(d), the GR values of the Bi-OWSP decline with an increase in job volume. Moreover, the effectiveness of Bi-OWSP is influenced by factors like minimizing task rejections that could have been successfully completed on each server and considering Frail Failure Tolerance (FFT). These factors play pivotal roles in improving the effectiveness of the Bi-OWSP approach.

We accomplished experiments using different numbers of virtual machines, as depicted in Figs. 7(a) and 7(b). Both metrics, energy and reliability, were divided into four containers, each containing virtual machines (VMs) with varying computing capabilities. A batch of application tasks was randomly scheduled across each container. The influx of jobs varies continuously, leading to fluctuations in the volume of VMs during the task execution. To assess the influence of these approaches compared to the ECR, as depicted in Fig. 7(a), the ECR values were recorded every five seconds. The ECR values of Bi-OWSP and EQ-VMC exhibited a relatively similar pattern. As the second container received a large influx of jobs, the ECR increased for all algorithms due to a significant rise in VM overhead. Conversely, as the number of tasks decreased, the Bi-OWSP algorithm closed many idle VMs to improve the ECR, surpassing similar algorithms in performance.

In Fig. 7(b), we compare the algorithms (Bi-OWSP, ReadFS, FDRGS, and SERAS) based on the reliability metric. The trend of the Bi-OWSP algorithm demonstrates efficient performance. However, the fluctuation in reliability observed in the latter two algorithms (FDRGS and SERAS) is more pronounced compared to Bi-OWSP and ReadFS. We can observe that the reliability rate of Bi-OWSP and ReadFS consistently remains higher than that of FDRGS and SERAS. This is because the former algorithms take resource failure into consideration, whereas the latter algorithms do not emphasize it sufficiently. As a result, some scheduled tasks are rejected. Although this rejection may reduce resource consumption, it weakens the overall outcomes for the service providers.

Fig. 8 presents a comparison of energy and reliability trade-offs among the Bi-OWSP, ReadFS, EQ-VMC, and FDRGS algorithms. The graph displays stacked bars for each algorithm, representing three groups of workflow applications with different data sizes.

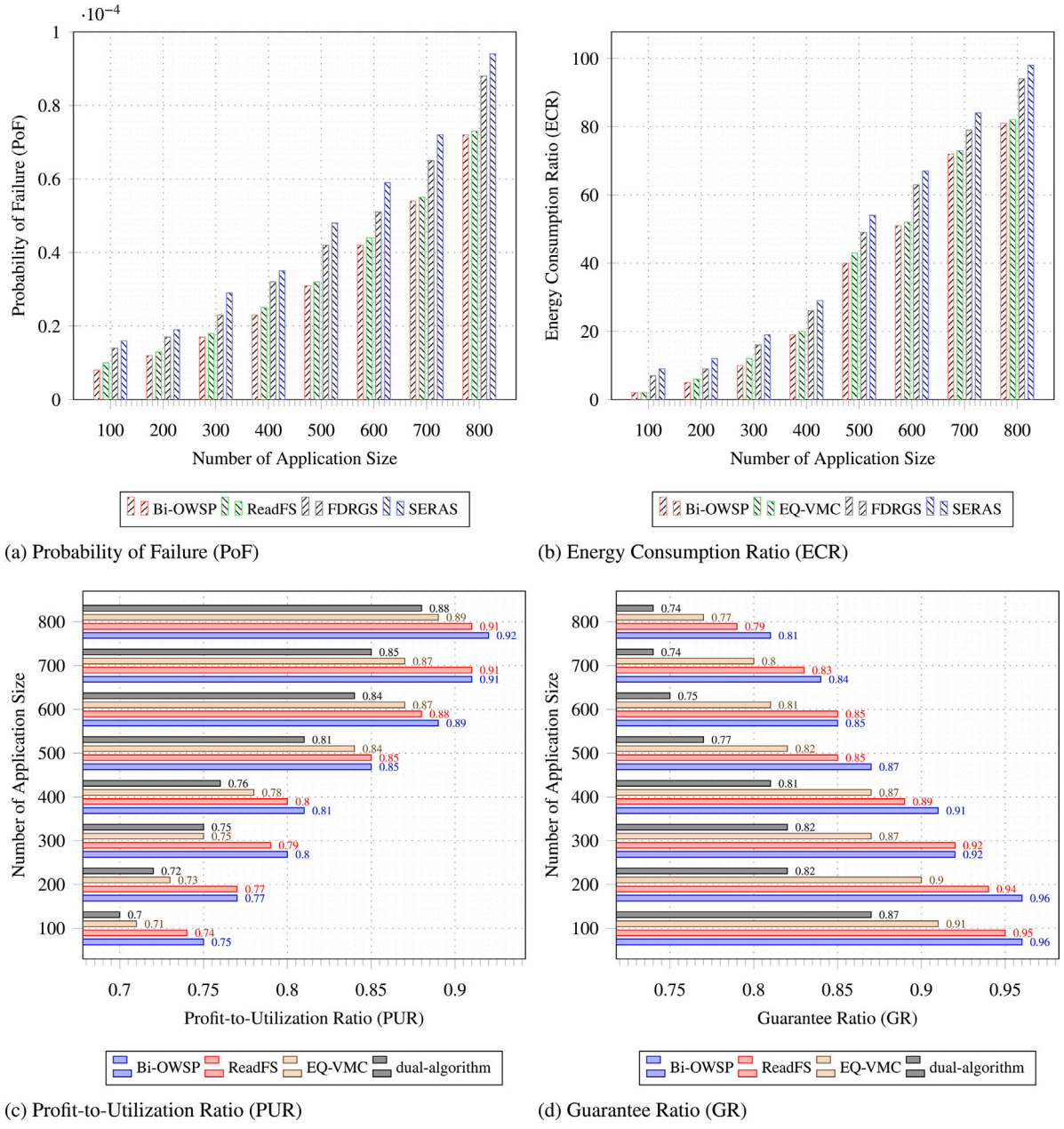
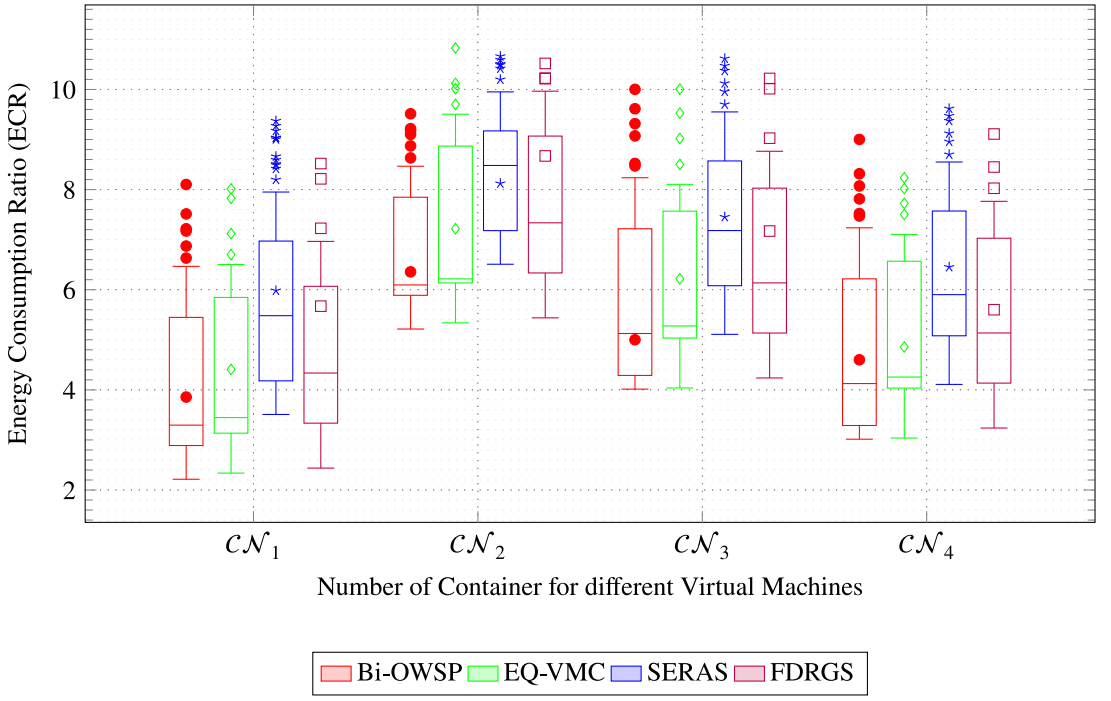
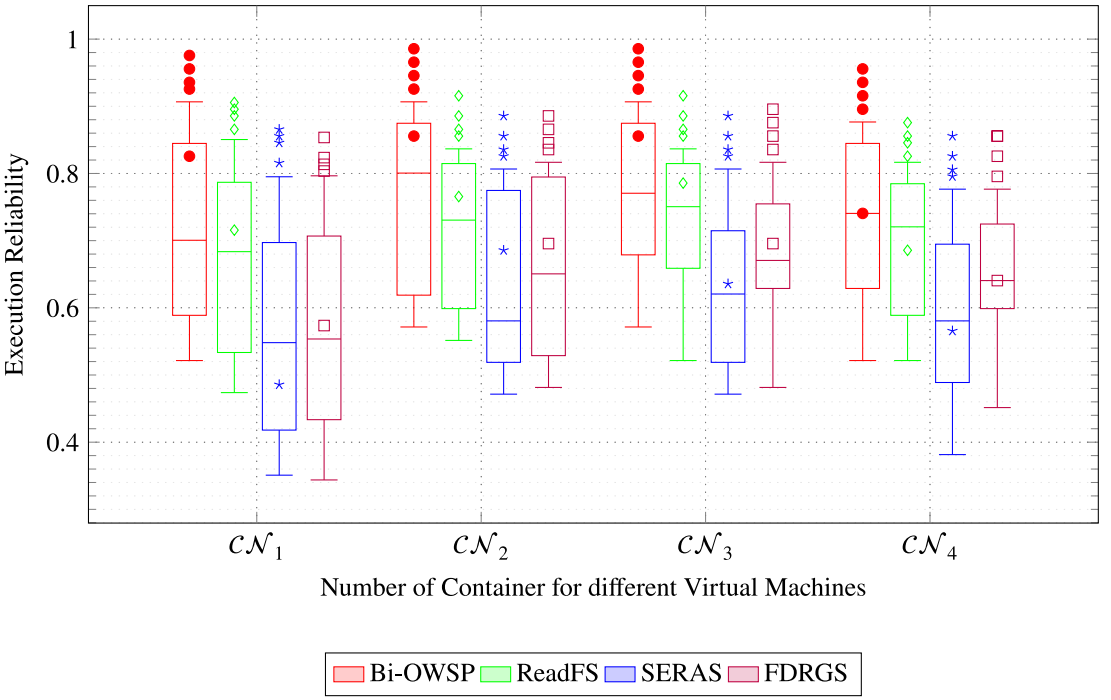


Fig. 6. Synthesized workflows: Average PoF, ECR, PUR, and GR as increased in number of applications sizes.

Each bar corresponds to a specific batch of applications. In Fig. 8, it is noteworthy that the ERT (Energy and Reliability Trade-off) initially experiences a rapid decrease for all algorithms. Subsequently, it gradually increases and eventually tends to achieve a better balance between the two metrics. This indicates that when there are fewer active servers in the system, the algorithms struggle to accommodate new applications. However, as the number of high-performance hosts rises, a larger proportion of accepted tasks can be accomplished successfully within the designated time constraints. As illustrated in Fig. 8, the ERT of the Bi-OWSP and ReadFS algorithms exhibits a significant increase initially, followed by a gradual convergence towards the optimum rate. This is because the Bi-OWSP algorithm incorporates the Frail Fault Tolerance (FFT) technique and considers resource failure, while the ReadFS algorithm also accounts for the probability of resource failure. When a large number of applications arrive simultaneously, these algorithms effectively schedule them, even if some of them fall within the realm of FFT. Consequently, the Bi-OWSP and ReadFS algorithms demonstrate greater robustness compared to other competing algorithms.



(a) The box plot shows the distribution of values, including the median, quartiles, and outliers, to provide a visual understanding of the data of energy cost



(b) The box plot presents a visual representation of the distribution of execution cost values

Fig. 7. The plot illustrates how the number of applications relates to energy expense ratio and reliability ratio of the synthesized workflows.

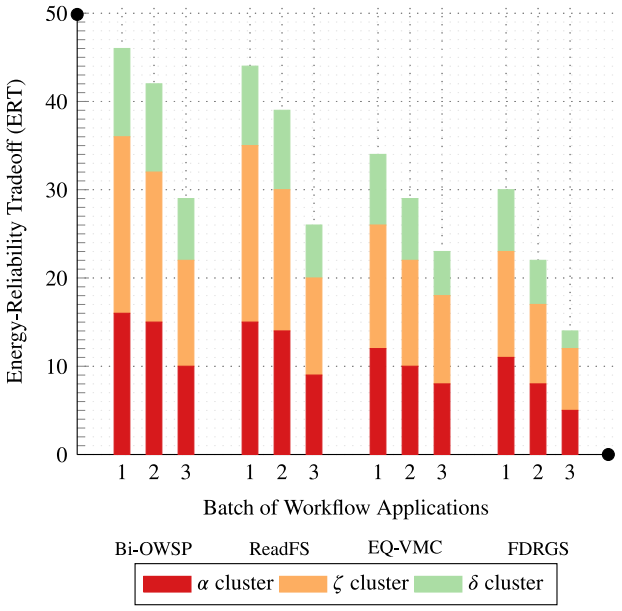


Fig. 8. Energy-Reliability Trade-off (ERT) comparison among algorithms.

7.2. Real scientific workflow applications

We conducted simulation experiments to assess how well our algorithm performs by creating two scientific workflows that mimic real-world scenarios. These workflows, namely Epigenomics and LIGO, require extensive computational analysis. The LIGO Inspiral Analysis Workflow, which focuses on detecting gravitational waves, comprises 100 sub-workflows and over 200,000 application modules, as documented in earlier studies. A substantial number of these tasks are computationally demanding and hold high priority. On the other hand, Epigenomics is a biological field that employs multiple pipelines to sequence genomes, necessitating a considerable amount of computational resources.

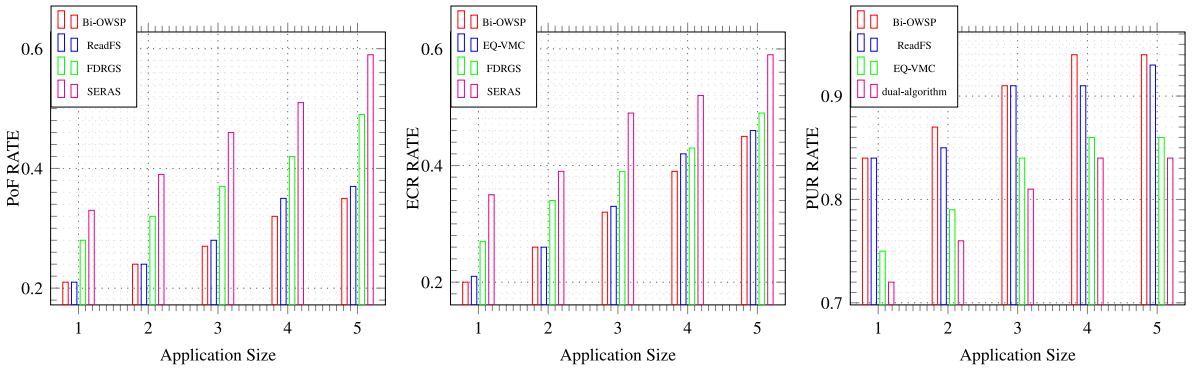
In order to simulate scientific workflow applications that closely resemble real-world scenarios in Epigenomics and LIGO, we employed the DAX (DAG in XML) format. Using this format, we were able to generate a diverse set of applications with different tasks, dependencies, and computational complexities. Our simulation specifically focused on five different sizes of workflow applications: small (approximately 50 applications), small-medium (around 100 applications), medium (about 200 applications), large (about 400 applications), and extra-large (about 600 applications). To ensure the robustness of our findings, we performed the experiments 15 times for each size and then calculated the average results for comparison. In this section, we primarily concentrated on three important metrics used to assess the efficiency of the applications. These metrics include the energy cost ratio, the failure probability, and the performance-to-usage ratio. We analyzed these metrics across all five application sizes, enabling a comprehensive assessment of performance.

7.2.1. Results of comparing different application sizes

For the upcoming experiment, the probability of failure model (FP) was used to generate values of 0.35 for the application's execution risk rate and 0.1 for the probability of failure parameter. The analysis for the Epigenomics and LIGO simulation results is carried out on a varying number of applications as S (1), S-M (2), M (3), L (4), and E-L (5), where they indicate small, small-medium, medium, large, and extra-large sizes workflow applications, respectively.

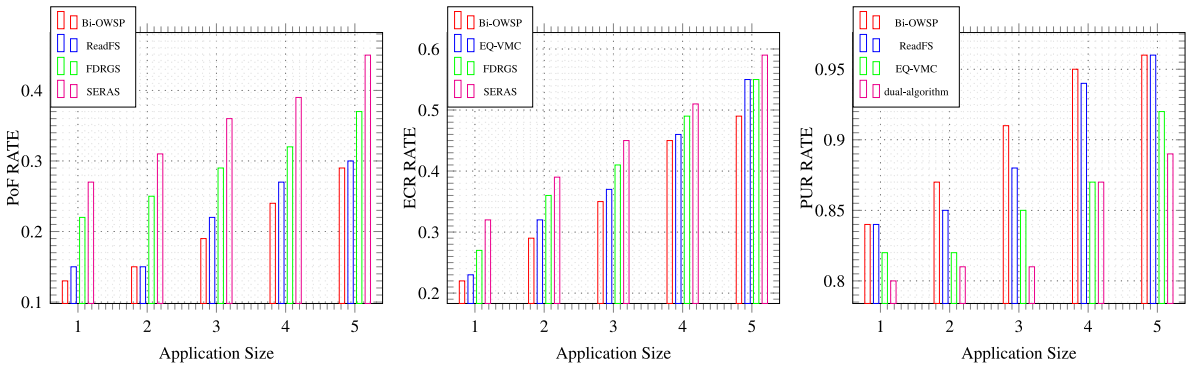
Fig. 9 illustrates the experimental outcomes of scheduling scientific Epigenomics applications across various application sizes. The comparison involves four algorithms: Bi-OWSP, ReadFS, FDRGS, and SERAS. From the results shown in Fig. 9, it is evident that Bi-OWSP and ReadFS outperform the other two algorithms in terms of achieving higher reliability by reducing the Probability of Failure (PoF) rate. These algorithms have advanced features that take into account the failure rates of communication connections. They exclude processing servers that are more prone to failure within a specified reference period, disregarding host applications whose likelihood of failure exceeds predefined limits.

Fig. 9(b) depicts that the Bi-OWSP algorithm outperforms other compared algorithms, including EQ-VMC, FDRGS, and SERAS, in terms of energy savings. The Bi-OWSP algorithm achieves energy savings in the range of approximately 12% to 27%. This reduction in operational costs can be attributed to the algorithm's ability to effectively adjust frequencies using the DVFS mechanism. Specifically, the Bi-OWSP algorithm intelligently selects the optimal frequency for task execution, resulting in higher Profit-to-Utilization Ratio (PUR) values. Moreover, the algorithm employs a resource allocation strategy that efficiently aggregates applications into processing servers, utilizing a task reclamation approach to optimize resource allocation.



(a) Probability of Failure (PoF) (b) Energy Consumption Ratio (ECR) (c) Profit-to-Utilization Ratio (PUR)

Fig. 9. The Epigenomics simulation outcomes with different application size.



(a) Probability of Failure (PoF) (b) Energy Consumption Ratio (ECR) (c) Profit-to-Utilization Ratio (PUR)

Fig. 10. The LIGO experimental results with different application size.

Fig. 9(c) presents a comparison between the Bi-OWSP algorithm and ReadFS, EQ-VMC, and the dual-algorithm approaches in terms of the PUR metric. It is evident from the figure that both Bi-OWSP and ReadFS outperform EQ-VMC and the dual-algorithm approaches across all application size scenarios. The advantage of the former algorithms can be attributed to their sophisticated control means, which efficiently prevent virtual machines from hosting applications and minimize resource wastage. As a result, both Bi-OWSP and ReadFS exhibit high PUR values that are relatively close to each other.

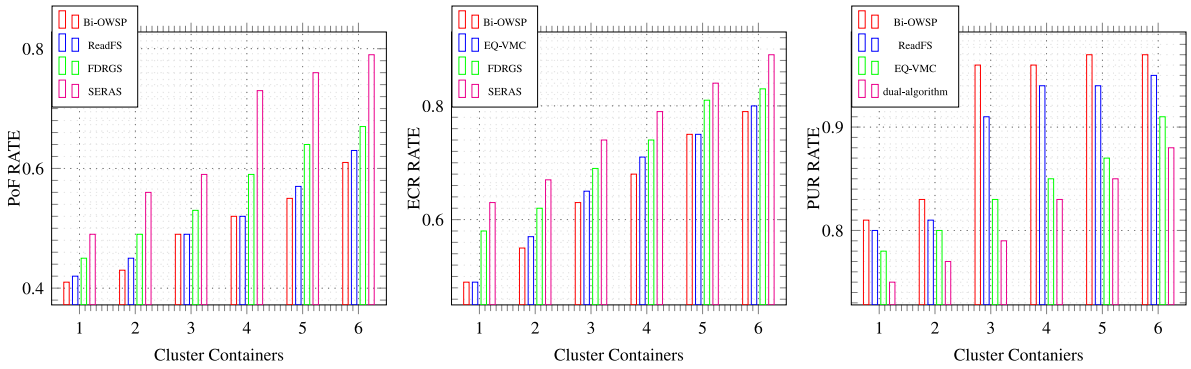
We performed identical experiments on the scientific LIGO application, employing the same evaluation metrics and proposed algorithms, as depicted in Fig. 10. Fig. 10(a) unveils that the Bi-OWSP and ReadFS algorithms exhibit the highest average tolerance, indicating their robustness and ability to withstand additional failures. The Bi-OWSP policy demonstrates slightly higher tolerance than the ReadFS algorithm, surpassing the FDRGS algorithm by 2.7% and the SERAS algorithm by 29.7%. However, it is 38.3% higher than the SERAS algorithm. The Bi-OWSP heuristics employ a selection strategy that identifies robust resources with low Probability of Failure (PoF). Conversely, the FDRGS and SERAS algorithms exhibit certain flaws that adversely affect the PoF metric.

In the majority of cases, the Bi-OWSP and EQ-VMC policies outperform other energy-saving schemes, as depicted in Fig. 10(b). In terms of Energy Consumption Ratio (ECR), these algorithms exhibit superior performance compared to the FDRGS and SERAS methods. It is also observed that ECR increases as the application sizes grow. However, the Bi-OWSP algorithm consistently demonstrates its superiority and robustness, achieving the highest energy savings across different application sizes. The stability of the Bi-OWSP and EQ-VMC policies surpasses that of the FDRGS and SERAS algorithms, even when subjected to varying workloads.

Fig. 10(c) demonstrates an increase in PUR for the Bi-OWSP and ReadFS approaches compared to other comparable methods. This improvement becomes particularly significant as the number of application sizes increases. Additionally, it is observed that as application sizes grow, the variation between algorithms becomes more pronounced. The Bi-OWSP policy excels in minimizing resource wastage by effectively placing applications, thereby consistently reducing resource leakage. The experiments conducted highlight the ability of our algorithm to enhance resource utilization and improve system productivity.

7.2.2. Results of comparing different server capacity

For this study, we chose a moderate-sized Epigenomics and LIGO application to compare. We found that Bi-OWSP demonstrated similar enhancements in achieving a more favorable combination of reliability and energy consumption objectives. The server is



(a) Probability of Failure (PoF) (b) Energy Consumption Ratio (ECR) (c) Profit-to-Utilization Ratio (PUR)

Fig. 11. The Epigenomics experimental results with different server capacity.

arranged in a descending order based on the number of virtual machines and their computing capacity. Additionally, the virtual machines are divided into two categories. The initial group comprises 75 virtual machines, namely DL-1 (1), DL-2 (2), and DL-3 (3), while the second group comprises 120 virtual machines, namely DL-4 (4), DL-5 (5), and DL-6 (6).

Fig. 11 illustrates the outcomes of the experiment involving the placement of an Epigenomics application on various virtual machines. Fig. 11(a) demonstrates that when multiple applications are directed to each cluster server, the Bi-OWSP algorithm optimizes the allocation of virtual machines to fulfill scheduling requirements while taking the Frail Fault Tolerance (FFT) approach into account. As a result, it exhibits a lower Probability of Failure (PoF) compared to alternative algorithms. The PoF values of the three algorithms are somewhat comparable. Contrarily, when the number of tasks is descending, the Bi-OWSP approach closes some idle virtual machines to improve the ECR, as shown in Fig. 11(b). Nonetheless, the decrease in the number of virtual machines is not immediate but rather involves a delay as the task necessitates a certain duration to execute after being scheduled, and this time interval corresponds to the delay. We are therefore seeing a steady reduction in ECR from DL-6 to DL-1. Comparing Bi-OWSP and EQ-VMC, shown in Fig. 11(b), both algorithms achieve reasonable ECR, but the Bi-OWSP is significantly higher than the SERAS, especially in DL-4, DL-5, and DL-6. This experimental observation indirectly highlights the benefits of implementing failure rate strategies for servers and communication links. By extensively utilizing these strategies, energy consumption can be effectively reduced.

Fig. 11(c) compares PUR between Bi-OWSP and other competing algorithms. It demonstrates a consistent trend where Bi-OWSP consistently maintains a higher PUR. By comparing Bi-OWSP to ReadFS (as shown in Fig. 11(c)), we can observe that both Bi-OWSP and ReadFS consistently outperform FDRGS and SERAS in terms of PUR. The superior PUR achieved by Bi-OWSP can be attributed to its more significant optimization of resource leakage compared to other algorithms.

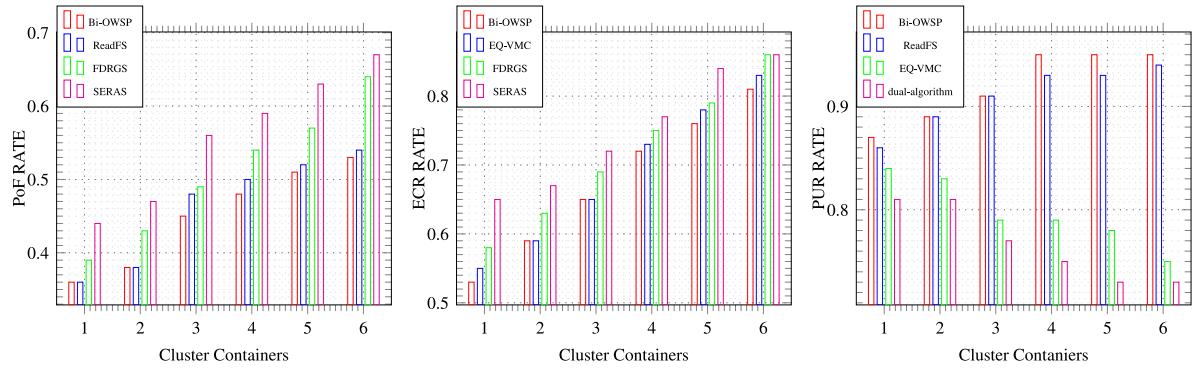
Experiments were conducted on scientific LIGO applications to confirm the efficacy of our suggested algorithm, as depicted in Fig. 12. The changes in PoF as the number of VM increases over time are shown in Fig. 12(a). The Bi-OWSP's PoF is lower than that of the other three algorithms. This means that many applications should be assigned enough virtual machines when directed to a cloud system. Increased virtual machines should satisfy the tolerance of fragile failures. The POF level rises by neglecting other algorithms' tolerance to fragile failures. On the contrary, Bi-OWSP will eventually complete the most accepted applications on time as Bi-OWSP considers FFT approaches.

On the other hand, Fig. 12(b) shows that the Bi-OWSP is very persistent from the point of view of ECR. This means that the algorithm is more stable in managing the short-term running energy consumption, and many idle servers that could have been initiated are excluded. Therefore, Bi-OWSP is a more robust algorithm in comparison to other algorithms.

The Bi-OWSP algorithm demonstrates outstanding performance concerning PUR, as depicted in Fig. 12(c). The experimental results indicate that all four algorithms exhibit stable PUR values, with Bi-OWSP demonstrating the highest performance. However, alternative heuristics like FDRGS and SERAS are unable to accept applications belonging to the Frail Fault Tolerance (FFT) category, leading to the rejection of certain applications that could have been successfully completed during application placement. As a result, the PUR of these heuristics decreases, negatively impacting their overall performance.

7.3. Statistical analysis

Table 4 displays the percentage enhancement in the performance of heuristics Bi-OWSP and ReadFS when compared to similar algorithms across all metrics. The initial section of Table 4 presents the Probability of Failure (PoF) values for various algorithms under different workloads, with Bi-OWSP and ReadFS demonstrating superior performance. Initially, for workloads ranging from S to M-L, the PoF values of all algorithms are relatively similar. However, as the workloads increase from M-L to E, the differences in PoF among algorithms fluctuate significantly. Comparing Bi-OWSP to other algorithms, it is evident that Bi-OWSP consistently maintains a lower PoF. This is because Bi-OWSP takes into account the Frail Fault Tolerance (FFT), enabling the successful scheduling



(a) Probability of Failure (PoF) (b) Energy Consumption Ratio (ECR) (c) Profit-to-Utilization Ratio (PUR)

Fig. 12. The LIGO experimental results with different server capacity.

of numerous tasks without violating deadline constraints. Neglecting FFT may raise the PoF but would also undermine algorithm performance.

The energy usage of the competing algorithms is depicted in the second part of Table 4. In general, the E and E-L instance types exhibit higher energy consumption compared to the other five instance types, primarily due to the significant number of assigned tasks. Remarkably, the Bi-OWSP and EQ-VMC algorithms outperform the others in terms of the ECR. This is attributed to the fact that these algorithms consider the residual energy consumption of cloud servers and exclusively assign tasks to servers with an adequate energy supply for processing operations.

The Table 4 also includes the measurement of another criterion known as the Profit-to-Utilization Ratio (PUR). It indicates that the Bi-OWSP, ReadFS, and EQ-VMC algorithms exhibit significantly higher PUR values compared to the dual-algorithm. These algorithms have been suitably adjusted to minimize resource leakage by employing clique partitioning techniques, thereby reducing virtual machine (VM) overhead. A higher PUR signifies increased system efficiency, as it indicates that cloud resources are effectively utilized for application management.

Table 4 provides insights into the guarantee ratio of scheduled tasks, which reaches saturation as the number of directed tasks increases. The rejection ratio for instance types S to M-L is relatively low. The findings indicate that the rejection ratio of tasks gradually increases with a higher task count. In the third part of Table 4, it is evident that the proposed Bi-OWSP algorithm exhibits a significantly lower rejection ratio compared to other algorithms. Both the EQ-VMC and dual-algorithm exhibit higher rejection ratios than Bi-OWSP. This disparity can be attributed to the fact that Bi-OWSP incorporates a fault-tolerant mechanism and a classification-based strategy to assign tasks to the most suitable servers, while other algorithms may reject a considerable number of applications that could have been successfully executed.

In the final section of Table 4, the energy-reliability trade-off between Bi-OWSP and the comparative algorithm is examined. Unlike the other algorithms, the Bi-OWSP algorithm achieves the highest offset value for all instances.

8. Conclusion and future work

The primary objective of this paper is to enhance energy efficiency and reliability in diverse cloud environments. In traditional research, the consideration for transient failures has been lacking. However, to meet the demands of priority constraints in large-scale cloud computing, designing highly reliable application placements has become essential. Therefore, we have developed an application paradigm tailored to directed acyclic graphs (DAGs) to improve task reliability while achieving energy savings.

The proposed algorithm operates through three sub-algorithms. Firstly, the stepwise dynamic voltage and frequency scale algorithm is employed to calculate the optimal frequency for each server, thereby reducing energy consumption. Secondly, the reliability-conscious heterogeneous fault tolerance algorithm is adjusted to prevent the utilization of servers and communication links with high-deficits, ensuring improved reliability. Lastly, a novel energy-aware stepwise reliability maximization algorithm is formulated to determine the optimal combination of tasks and servers. This algorithm considers both energy depreciation and reliability maximization as crucial factors, ultimately leading to the selection of the most suitable task-server combinations. The evaluation of various metrics confirms that the Bi-OWSP algorithm surpasses other algorithms' performance, as demonstrated by the simulation results.

One potential future research direction is to develop methods for detecting and handling failed applications in a multiprocessor computing system with varying utilization levels. This could lead to optimized processor frequency utilization and improved multi-dimensional resource partitioning, reducing resource fragmentation and ensuring high system reliability for heterogeneous computing clusters.

Table 4
Average five evaluated metrics against different application sizes.

Paradigms	Probability of failure						
	S	S-M	M	M-L	L	L-E	E
Bi-OWSP	0.05	0.08	0.16	0.21	0.34	0.51	0.67
ReadFS	0.05	0.08	0.17	0.23	0.34	0.53	0.67
FDRGS	0.08	0.09	0.21	0.29	0.36	0.61	0.74
SERAS	0.11	0.12	0.27	0.34	0.42	0.65	0.81
Paradigms	Energy consumption ratio						
	S	S-M	M	M-L	L	L-E	E
Bi-OWSP	3.21	05.04	09.68	14.82	19.46	24.57	31.07
EQ-VMC	3.45	05.74	10.02	15.45	19.48	24.89	33.64
FDRGS	5.64	09.74	19.37	28.07	36.45	42.15	51.78
SERAS	9.67	13.28	17.64	34.51	41.08	51.64	59.44
Paradigms	Profit-to-Utilization ratio						
	S	S-M	M	M-L	L	L-E	E
Bi-OWSP	0.78	0.81	0.85	0.89	0.93	0.95	0.95
ReadFS	0.77	0.81	0.84	0.89	0.91	0.94	0.94
EQ-VMC	0.75	0.80	0.81	0.86	0.88	0.89	0.92
dual-algorithm	0.73	0.77	0.79	0.85	0.85	0.86	0.89
Paradigms	Guarantee ratio						
	S	S-M	M	M-L	L	L-E	E
Bi-OWSP	0.94	0.94	0.93	0.91	0.87	0.85	0.83
ReadFS	0.93	0.93	0.92	0.87	0.86	0.85	0.81
EQ-VMC	0.91	0.90	0.88	0.85	0.82	0.81	0.79
dual-algorithm	0.89	0.88	0.87	0.84	0.81	0.79	0.77
Paradigms	Energy-Reliability trade-off						
	S	S-M	M	M-L	L	L-E	E
Bi-OWSP	0.81	0.84	0.84	0.86	0.88	0.91	0.91
ReadFS	0.79	0.81	0.84	0.85	0.87	0.91	0.91
EQ-VMC	0.78	0.80	0.80	0.81	0.85	0.89	0.90
FDRGS	0.76	0.78	0.78	0.80	0.82	0.83	0.85

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article

References

[1] C.D. Anisha, K.G. Saranya, An intense study on intelligent service provisioning for multi-cloud based on machine learning techniques, EAI/Springer Innov. Commun. Comput. (2022) 177–200, http://dx.doi.org/10.1007/978-3-030-74402-1_10/COVER/, URL https://link.springer.com/chapter/10.1007/978-3-030-74402-1_10.

[2] R. Shaw, E. Howley, E. Barrett, Applying reinforcement learning towards automating energy efficient virtual machine consolidation in cloud data centers, Inf. Syst. 107 (2022) 101722, http://dx.doi.org/10.1016/J.IS.2021.101722.

[3] S. Tuli, S.S. Gill, M. Xu, P. Garraghan, R. Bahsoon, S. Dustdar, R. Sakellariou, O. Rana, R. Buyya, G. Casale, N.R. Jennings, HUNTER: AI based holistic resource management for sustainable cloud computing, J. Syst. Softw. 184 (2022) 111124, http://dx.doi.org/10.1016/J.JSS.2021.111124.

[4] Y. Cai, J. Llorca, A.M. Tulino, A.F. Molisch, Ultra-reliable distributed cloud network control with end-to-end latency constraints, IEEE/ACM Trans. Netw. (2022) 1–16, http://dx.doi.org/10.1109/TNET.2022.3179349.

[5] A.K. Bhoi, M.R. Kabat, S.C. Nayak, G. Palai, Energy efficient task allocation and consolidation in multicast cloud network, in: Wireless Networks 2022, 2022, pp. 1–18, http://dx.doi.org/10.1007/S11276-022-03029-2, URL https://link.springer.com/article/10.1007/s11276-022-03029-2.

[6] M.K. Gupta, A. Jain, T. Amgoth, Power and resource-aware virtual machine placement for iaas cloud, Sustain. Comput.: Inform. Syst. 19 (2018) 52–60, http://dx.doi.org/10.1016/J.SUSCOM.2018.07.001.

[7] R. Mahmud, K. Ramamohanarao, R. Buyya, Latency-aware application module management for fog computing environments, ACM Trans. Internet Technol. (TOIT) 19 (2018) http://dx.doi.org/10.1145/3186592, URL https://dl.acm.org/doi/abs/10.1145/3186592.

[8] S.S. Nabavi, S.S. Gill, M. Xu, M. Masdari, P. Garraghan, TRACTOR: Traffic-aware and power-efficient virtual machine placement in edge-cloud data centers using artificial bee colony optimization, Int. J. Commun. Syst. 35 (2022) e4747, http://dx.doi.org/10.1002/DAC.4747, URL https://onlinelibrary.wiley.com/doi/full/10.1002/dac.4747https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4747https://onlinelibrary.wiley.com/doi/10.1002/dac.4747.

[9] M.A. Khan, An efficient energy-aware approach for dynamic VM consolidation on cloud platforms, in: Cluster Computing 2021, Vol. 24, 2021, pp. 3293–3310, http://dx.doi.org/10.1007/S10586-021-03341-0, 424, URL https://link.springer.com/article/10.1007/s10586-021-03341-0.

[10] J. Zhou, K. Cao, X. Zhou, M. Chen, T. Wei, S. Hu, Throughput-conscious energy allocation and reliability-aware task assignment for renewable powered in-situ server systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 41 (2022) 516–529, http://dx.doi.org/10.1109/TCAD.2021.3068095.
[11] Y. Ramzanpoor, Mirsaeid, H. Shirvani, M. Golsorkhtabaramiri, Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure, in: Complex & Intelligent Systems 2021, Vol. 8, 2021, pp. 361–392, http://dx.doi.org/10.1007/S40747-021-00368-Z, 8:1 URL https://link.springer.com/article/10.1007/s40747-021-00368-z.
[12] C.K. Swain, N. Saini, A. Sahu, Reliability aware scheduling of bag of real time tasks in cloud environment, in: Computing 2019, Vol. 102, 2019, pp. 451–475, http://dx.doi.org/10.1007/S00607-019-00749-W, 102:2 URL https://link.springer.com/article/10.1007/s00607-019-00749-w.
[13] S. Kanwal, Z. Iqbal, F. Al-Turjman, A. Irtaza, M.A. Khan, Multiphase fault tolerance genetic algorithm for vm and task scheduling in datacenter, Inf. Process. Manage. 58 (2021) 102676, http://dx.doi.org/10.1016/J.IPM.2021.102676.
[14] R. Rani, R. Garg, Reliability aware green workflow scheduling using ϵ-fuzzy dominance in cloud, in: Complex & Intelligent Systems 2021, Vol. 8, 2021, pp. 1425–1443, http://dx.doi.org/10.1007/S40747-021-00609-1, 8:2 URL https://link.springer.com/article/10.1007/s40747-021-00609-1.
[15] R. Medara, R.S. Singh, Energy efficient and reliability aware workflow task scheduling in cloud environment, in: Wireless Personal Communications 2021, Vol. 119, 2021, pp. 1301–1320, http://dx.doi.org/10.1007/S11277-021-08263-Z, 119:2 URL https://link.springer.com/article/10.1007/s11277-021-08263-z.
[16] H.A. Hassan, S.A. Salem, E.M. Saad, A smart energy and reliability aware scheduling algorithm for workflow execution in DVFS-enabled cloud environment, Future Gener. Comput. Syst. 112 (2020) 431–448, http://dx.doi.org/10.1016/J.FUTURE.2020.05.040.
[17] R. Garg, M. Mittal, L.H. Son, Reliability and energy efficient workflow scheduling in cloud environment, in: Cluster Computing 2019, Vol. 22, 2019, pp. 1283–1297, http://dx.doi.org/10.1007/S10586-019-02911-7, 22:4 URL https://link.springer.com/article/10.1007/s10586-019-02911-7.
[18] J. Yao, N. Ansari, Fog resource provisioning in reliability-aware IoT networks, IEEE Internet Things J. 6 (2019) 8262–8269, http://dx.doi.org/10.1109/JIOT.2019.2922585.
[19] Z. Li, V. Chang, H. Hu, H. Hu, C. Li, J. Ge, Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds, Inform. Sci. 568 (2021) 13–39, http://dx.doi.org/10.1016/J.INS.2021.03.003.
[20] W. Lin, G. Peng, X. Bian, S. Xu, V. Chang, Y. Li, Scheduling algorithms for heterogeneous cloud environment: Main resource load balancing algorithm and time balancing algorithm, J. Grid Comput. 17 (2019) 699–726, http://dx.doi.org/10.1007/S10723-019-09499-7/METRICS, URL https://link.springer.com/article/10.1007/s10723-019-09499-7.
[21] Z. Li, X. Yu, L. Yu, S. Guo, V. Chang, Energy-efficient and quality-aware VM consolidation method, Future Gener. Comput. Syst. 102 (2020) 789–809, http://dx.doi.org/10.1016/J.FUTURE.2019.08.004.
[22] X. Xia, H. Qiu, X. Xu, Y. Zhang, Multi-objective workflow scheduling based on genetic algorithm in cloud environment, Inform. Sci. 606 (2022) 38–59, http://dx.doi.org/10.1016/J.INS.2022.05.053.
[23] B. Zhao, H. Aydin, D. Zhu, On maximizing reliability of real-time embedded applications under hard energy constraint, IEEE Trans. Ind. Inform. 6 (2010) 316–328, http://dx.doi.org/10.1109/TII.2010.2051970.
[24] K. Li, Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers, IEEE Trans. Comput. 61 (2012) 1668–1681, http://dx.doi.org/10.1109/TC.2012.120.
[25] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, K. Li, Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster, Inform. Sci. 319 (2015) 113–131, http://dx.doi.org/10.1016/J.INS.2015.02.023.
[26] X. Tang, Reliability-aware cost-efficient scientific workflows scheduling strategy on multi-cloud systems, IEEE Trans. Cloud Comput. (2021) http://dx.doi.org/10.1109/TCC.2021.3057422.
[27] S. Ghanavati, J. Abawajy, D. Izadi, Automata-based dynamic fault tolerant task scheduling approach in fog computing, IEEE Trans. Emerg. Top. Comput. 10 (2022) 488–499, http://dx.doi.org/10.1109/TETC.2020.3033672.
[28] F. Cao, M.M. Zhu, Distributed workflow mapping algorithm for maximized reliability under end-to-end delay constraint, J. Supercomput. 2013 66 (2013) 1462–1488, http://dx.doi.org/10.1007/S11227-013-0938-3, 66:3 URL https://link.springer.com/article/10.1007/s11227-013-0938-3.
[29] X. Li, Z. Qian, S. Lu, J. Wu, Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center, Math. Comput. Modelling 58 (2013) 1222–1235, http://dx.doi.org/10.1016/j.mcm.2013.02.003.
[30] M.K. Gupta, T. Amgoth, Resource-aware virtual machine placement algorithm for iaas cloud, J. Supercomput. 74 (2018) 122–140, http://dx.doi.org/10.1007/S11227-017-2112-9/METRICS, URL https://link.springer.com/article/10.1007/s11227-017-2112-9.
[31] S.K. Garg, C.S. Yeo, A. Anandasivam, R. Buyya, Energy-efficient scheduling of HPC applications in cloud computing environments, 2009, http://dx.doi.org/10.48550/arxiv.0909.1146, URL https://arxiv.org/abs/0909.1146v1.
[32] M.C. Filho, R.L. Oliveira, C.C. Monteiro, P.R. Inácio, M.M. Freire, CloudSim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness, in: Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management, 2017, pp. 400–406, http://dx.doi.org/10.23919/INM.2017.7987304.
[33] E. Andrade, B. Nogueira, Performability evaluation of a cloud-based disaster recovery solution for IT environments, J. Grid Comput. 2018 17 (2018) 603–621, http://dx.doi.org/10.1007/S10723-018-9446-2, 17:3 URL https://link.springer.com/article/10.1007/s10723-018-9446-2.
[34] T. Goyal, A. Singh, A. Agrawa, Cloudsim: simulator for cloud computing infrastructure and modeling, Procedia Eng. 38 (2012) 3566–3572, http://dx.doi.org/10.1016/J.PROENG.2012.06.412.
[35] S. Ijaz, E.U. Munir, S.G. Ahmad, M.M. Rafique, O.F. Rana, Energy-makespan optimization of workflow scheduling in fog–cloud computing, in: Computing 2021, Vol. 103, 2021, pp. 2033–2059, http://dx.doi.org/10.1007/S00607-021-00930-0, 103:9 URL https://link.springer.com/article/10.1007/s00607-021-00930-0.
[36] A. Dogan, F. Özgüner, Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (2002) 308–323, http://dx.doi.org/10.1109/71.993209.
[37] J.S. Plank, W.R. Elwasif, Experimental assessment of workstation failures and their impact on checkpointing systems, in: Digest of Papers - 28th Annual International Symposium on Fault-Tolerant Computing, FTCS 1998, 1998, pp. 48–57, http://dx.doi.org/10.1109/FTCS.1998.689454.
[38] M. Mao, M. Humphrey, A performance study on the VM startup time in the cloud, in: Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012, 2012, pp. 423–430, http://dx.doi.org/10.1109/CLOUD.2012.103.
[39] R. Mahmud, S.N. Srirama, K. Ramamohanarao, R. Buyya, Profit-aware application placement for integrated fog–cloud computing environments, J. Parallel Distrib. Comput. 135 (2020) 177–190, http://dx.doi.org/10.1016/J.JPDC.2019.10.001.
[40] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.H. Su, K. Vahi, M. Livny, Pegasus: Mapping scientific workflows onto the grid, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 3165, 2004, pp. 11–20, http://dx.doi.org/10.1007/978-3-540-28642-4_2/COVER, URL https://link.springer.com/chapter/10.1007/978-3-540-28642-4_2.
[41] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P.J. Maechling, R. Mayani, W. Chen, R.F.D. Silva, M. Livny, K. Wenger, Pegasus, a workflow management system for science automation, Future Gener. Comput. Syst. 46 (2015) 17–35, http://dx.doi.org/10.1016/J.FUTURE.2014.10.008.