Workload aware autonomic resource management scheme using grey wolf optimization in cloud environment

Bhupesh Kumar Dewangan1 | Amit Agarwal2 | Tanupriya Choudhury3 | Ashutosh Pasricha4

1 School of Computer Science, University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India
2 School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India
3 Associate professor, Department of Informatics, University of Petroleum and Energy Studies, Dehradun, India
4 Schlumberger Pvt. Ltd., New Delhi, India

Correspondence
Bhupesh Kumar Dewangan, School of Computer Science, University of Petroleum and Energy Studies, Bidholi, Via Prem Nagar, Dehradun 248007, Uttarakhand.
Email: b.dewangan@ddn.upes.ac.in

Abstract
Autonomic resource management on cloud is a challenging task because of its huge heterogeneous and distributed environment. There are several service providers in the cloud to provide a different set of cloud services. These services are delivered to the clients through a cloud network, and it needs to satisfy the Quality-of-Service (QoS) requirements of users without affecting the Service Level Agreements. It can only manage through autonomic cloud resource managing frameworks. However, most of the existing frameworks are not much efficient for managing cloud resources because of the varied applications and environments of the cloud. To defeat such problems, this paper proposed the workload aware Autonomic Resource Management Scheme (WARMS) in the cloud environment. Initially, the clustering of cloud workloads is achieved by Modified Density Peak Clustering algorithm. Further, the workload scheduling process is done using fuzzy logic for cloud resource availability. The autonomic system uses Grey Wolf Optimization for virtual machine deployment to achieve optimal resource provisioning. The WARMS system focused on reducing the Service Level Agreement violation, cost, energy usage, and time, and providing better QoS. The simulation results of WARMS shows the system delivering the cloud services more efficiently by the minimized rate of violation and enhanced QoS.

1 | INTRODUCTION

The term “Cloud computing” refers to the on-demand accessibility of cloud resources anytime-anywhere for computation, information processing, and data storage without the direct intervention of human resources [1]. The crucial role of cloud computing is to offer various services to users over the internet. Instead of the local computers, the cloud service providers deliver computations in a virtualized manner, which is enlarged by the cloud computing model. SaaS (Software as a Service), PaaS (Platform as a service), and IaaS (Infrastructure as a Service) are the three basic computing services in cloud computing [2–4]. The valuable computing requirements such as storage, applications, servers, networks, and services are shared by enabling convenient, ubiquitous, on-demand network access of the Cloud computing business model [5].

The key functionalities of the cloud computing model are on-demand self-assistance, comprehensive access to network, pooling of resources, quick scalability, and service monitoring. The key problem in the Cloud paradigm is to permit users to obtain their facilities all over the world without SLA violation [6]. The on-demand share and distribution of the resource is also a vital part of the service provider, which can be achieved by an efficient resource allocation process. The most important benefit of resource allocation is to improve useful resource usage as well as limit the working costs. The virtualization technology provides more flexibility, which includes hardware virtualization along with CPU, memory, storage, and network [7, 8].

The resource management in the cloud is essential to regularize the resource allocation and service provisioning from the perspective of both service provider and cloud user. Normally, resource allocation straight away assigns a suitable workload to the cloud resources. Still, scheduling is required to allocate the resources in an optimal way when various categories of tasks with different workloads are received at the cloud. The conventional resource allocation schemes are concerned about QoS
parameters but fail to address the type of the task and category of workload [9, 10]. The additional allocation of resources does not assure the performance due to the unawareness of workload type. The CPU intensive applications only require an additional processing unit to accomplish the tasks, but the present resource allocation schemes can additionally add a set of other resources which does not require to run the specific applications. The over-allocation of resources minimizes resource utilization as well as affects the scalability of resources.

Moreover, due to the dynamic behaviour of the cloud environment, the need for the resources demanded by the cloud users and the number of resources allocated for processing are frequently changing over time [11, 12]. Hence, the autonomic cloud management system must adopt the dynamic behaviour of tasks and control the trade-off among task types and QoS parameters. To mitigate the above discussed issues, an intelligent autonomic resource management scheme must be developed for data centres to accomplish frequently changing task demands with the help of a limited available number of resources.

The distributed cloud environment consists of heterogeneous computing resources and different resource-intensive workloads. Therefore, clustering of workloads becomes significant as it can efficiently reduce the problems related to scheduling of the workloads. It can ease the process of scheduling the multiple workloads in distributed computing. Clustering is basically used in grouping of the objects exhibiting similar features. In conventional studies, K-means clustering technique is employed for the clustering process as it provides high performance in terms of grouping of objects, but the dynamic feature and the presence of large count of workloads make the traditional K-means clustering inefficient. Also the initial centroid point in this method is chosen randomly which may give inappropriate results. Moreover, the results of K-means clustering highly depend upon the initial centroid point and hence there will be a great influence in the solutions. The typical first-in-first-out (FIFO) and Fair-share scheduling techniques failed to effectively match the task to the hosting node. The proposed WARMS system uses a modified density peak clustering algorithm (MDPCA) for clustering which finds the cluster centre quickly by the help of kernel function further fuzzy logic is employed to schedule the workload. For selecting the optimized VM for executing the workload, GWO (Grey Wolf Optimization) is utilized; at this process, faulty or malicious VM in the network will be detected and removed.

The remainder of this paper is organized as follows: Related works on autonomic resource management are discussed in Section 2. A detailed description of our proposed autonomic resource management scheme is discussed in Section 3. The proposed schemes simulation results are demonstrated and discussed in Section 4. In Section 5, we provide a conclusion of this paper.

2 | RELATED WORKS

Some of the recent related works related to autonomic resource management are listed below:

Suresh et al. [13] presented a resource allocation method with different techniques. Right here, the KFCM algorithm becomes utilized to cluster to be had resources. In this method, the process of allotting tasks with all available resources is carried out with Modified Cloud Resource Provisioning Algorithm (MCRP). On MCRP framework, initially, the optimal cost had been selected by using the PSO algorithm. Then, the result from in step is utilized, and the resource had been allotted. The authors of this paper described that most of the traditional k-modes clustering techniques might have multiple varieties of destination clusters on its created cluster mode range.

Domanal et al. [14] presented a resource scheduling method with a HYBRID Bio-Inspired algorithm. Here, the researchers particularly centred on cataloguing the jobs using MPSO, the assignment of task, and dealing with the cloud resources with the use of MPSO, MCSO, and HYBRID Bio-inspired procedures. The time intervals of the jobs will stay steady even if multiple jobs arrive at the same periods. The resources needed to process have been allotted after correctly allotting the jobs to the corresponding VMs. The MPSO algorithm performs a crucial function in allotting the coming jobs to the VMs as successfully as available. Clustering depends on the variety of VMs taken for experimentation, and VMs should need to have sufficient resources for executing the customers’ requests and providing it to VMs, which are needed to control by MPSO.

Ghobaei-Arani et al. [15] presented a technique with the base of resource provisioning for the applications of cloud, which are providing various services. Here, different concepts of two techniques (Reinforcement Learning [RL] and autonomic computing) were integrated by the authors. Additionally, they utilized IBM’s control loop technique called MAPE for acquiring autonomic computing. The MAPE is a WARMs agent that perceives its environment with the usage of sensors. Along with those perceptions for establishing the activities to be performed within the surroundings, it has been processed constantly and controls the VMs, which are allotted to every service provider at unique time durations.

Marinescu et al. [16] suggested a self-managing framework for scaling the management schemes of cloud resources. This presented framework includes a market-based tool which acts as an agent for accomplishing the principles of resource managing systems of cloud and determines how to expand the system by making the management system to manage the resource automatically. Firstly, clear communication among the cloud consumer and provider will be assumed in this presented approach. At the run time of applications, the services are expressed as a workload herewith the nodes characterize the edges and present applications by distinguishing the service steps. The authors said that this process could greatly reduce the cost.

Sotiriadis et al. [17] presented a self-managing method for VM scheduling in a cloud environment. There, the researchers used a platform called Open Stack for public and private cloud establishment. From the actual structure of Open Stack, it chooses the host after the placement of VMs based on the offered memory before it overreaches the amount of VMs limit. These operations may exceed loads of utilized PMs and remain low RAM. The presented method is effective and adaptable to
the utilization of historical data of PMs and VMs. Additionally, the machine learning technique is utilized on this self-managing scheduling method for pattern extraction, and it is done by examining the data and permanent inspection.

Filelis-Papadopoulos et al. [18] extended a framework of cloud simulation for making it support self-managing and organizing abilities with cloud resources. From the presented framework, the simulation process is carried out for altering the situation of systems more dynamic on every time step, and this process is done with the base of time advance looping technique by combining the components of the systems. When comparing this proposed hierarchical-based VM placement approach to the classical management approach, the hierarchical-based approach delivered much correct decision while placing the VMs.

Zuo et al. [19] introduced a method with the base of self-adaptive threshold called Dynamic Weighted Load Evaluation Method (SDWM). Initially, for estimating the state of the cloud resources, they presented the dynamic evaluation indicators for accurate estimation. Here, the SDWM separates a load of resources into three different situations; they are, normal state, idle state, and overloaded state. At last, the researchers exploited their method as a model for energy evaluation for describing the energy level utilized for migrating the resources based on the request from users. This method earned excellent adaptability, and even the resources leave or enter dynamically.

For Efficient Resource Management, Zahoor et al. [36] portrayed a Cloud/Fog based Smart Grid Model. To improve the systems’ response time and to reduce delay, the Fog computing remains an aid, and hence in the SG domain, the resource management cloud-fog computing is used. Some of the load balancing algorithms such as ACO, PSO, and ABC were used with the proposed HABACO algorithm for optimal resource management in cloud and fog based environment. For load balancing, five algorithms are employed amongst an SG user’s requests and service providers. The proposed work mainly determines a hierarchical cloud–fog computing structure to offer several computing services for SG resource management.

Xiong et al. [37] developed a Cloud/Fog Computing Resource Management and Pricing for Blockchain Networks. The price-based computing resource management was investigated to support the providers of offloading mining tasks to cloud/fog in proof-of-work based public blockchain networks. To investigate the profit maximization of cloud/fog provider and the utility maximization of miners, a two-stage Stackelberg game model was adopted. Also, the ideal resource management system having the discriminatory and uniform pricing for the cloud/fog provider was implied and studied. The proposed analytical model was validated by performing the real experiment. The network performance is evaluated using simulations which aid the cloud/fog provider to gain the highest profit and to achieve the optimal resource management.

Gai et al. [38] presented a Heterogeneous Cloud Computing based Resource Management scheme in Sustainable Cyber-Physical Systems. The sustainability of the system was increased by combining CPS with a heterogeneous cloud computing method. The task assignment is considered as an NP-hard problem in this paper. Tasks are assigned to the heterogeneous clouds in the Smart Cloud-based Optimizing Workload (SCOW) model using predictive cloud capacities. The competitiveness of the enterprises is embarrassed due to unstable service demands. The Workload Resource Minimization Algorithm (WRM), Smart Task Assignment (STA) Algorithm, and Task Mapping Algorithm (TMA) algorithms were proposed to reach the optimization objective.

The following Table 1 summarizes the recent autonomous resource provisioning works; their techniques, objective, performance metrics, advantage, and disadvantages are discussed.

The significant capability of a VM deployment algorithm is to balance the workloads to the resources for optimal performance. So, varied task parameters are considered for VM deployment. Available resources should be effectively utilized to achieve fair resource utilization. As users increase, tasks to be scheduled also increase. So, better algorithms are needed to deploy VM on systems. VM deployment algorithms are service-oriented and vary in environments. Meta-Heuristic optimization algorithms are employed to improve the effectiveness or to reduce the cost of the scheduling process. To reduce the computational cost, most of the scheduling problems use various optimization techniques. Meta-heuristic algorithms like PSO and Simulated Annealing are powerful methods for solving many optimization problems. Apart from that, GWO algorithm is a well-recognized optimization technique to resolve the various non-linear problems in numerous research areas due to its well suitable parameter settings and to achieve the optimal solution in a minimum number of iterations.

3 | DESIGN OF WARMS ARCHITECTURE

The architectural design of WARMS comprises three different phases, that is, workload submission, optimal resource provisioning, and service monitoring. In the workload submission phase, cloud users submit the works based on their demand. An optimal resource provisioning phase manages the cloud resources, and the workload is coming from the cloud users more efficiently with the help of clustering and VM deployment. Service Monitoring phase maintains the QoS parameters and self-management properties of Service Level Agreement (SLA). The proposed WARMS architecture is shown below.

a. **Cloud Users:** The users are the consumers present under the provider, who request for the specific resource based on their requirements. The requests collected from the users are considered as the workload and used for further process.

b. **Cloud Provider:** The cloud provider offers a set of services and manages the resources that are stored on their database. These resources will be distributed based on the requirements of the users.

c. **QoS Management:** It manages the possible QoS requirements to access the workloads. Based on the QoS needs of the workload, it will be managed.
TABLE 1 Comparison of autonomic resource provisioning techniques

| Technique | Objective | Performance metrics | Advantage | Disadvantage |
|-----------|-----------|---------------------|-----------|--------------|
| Type-2 fuzzy logic based auto scaling | To develop an elasticity Controller for autonomic resource provisioning | Workload estimation accuracy, effectiveness, robustness | Decrease cost of ownership without violating SLAs | Task type and their relationship with resource demands does not consider |
| Jamshidi et al. [29] | | | | |
| Auto-scaling mechanism based on machine learning | To accurately predict the processing load of a distributed server and optimize the service response time | mean absolute error, mean squared error and root mean squared error | Minimize the service time, and fulfill the SLA contracted by the user | The problem of starvation cannot be reduced |
| Moreno-Vozmediano et al. [30] | | | | |
| Reinforcement learning-based elasticity Controller | To scale up quickly to meet rising demand but also scale down by shutting down excess servers | Number of SLA violations, overall cost of the system | Reduce the cost of provisioning infrastructure | Self-protecting and self-healing attributes are not considered |
| Nouri et al. [31] | | | | |
| Learning automata-based resource provisioning | Dynamic resource provisioning | Response time, cost and allocated virtual machines | To reduce the cost of resource rental for the cloud gaming provider | Self-protecting is not considered |
| Aslanpouret al. [32] | | | | |
| Self-configuration and self-healing in resource management | To enhance the quality of cloud services | Execution cost, resource contention, execution time and SLA violation | Improves user satisfaction by maintaining SLA based on QoS requirements | Starvation problem exists |
| Gill et al. [21] | | | | |
| Reinforcement learning-based Proactive Auto-Scalar Benifa and Dejey [33] | To allocate necessary resources by learning the environment dynamically. | CPU utilization, response time and throughput | Learn the environment in parallel which is utilized for resource allocation in a realistic cloud environment | Self-protecting and self-healing attributes are not considered |
| KNN-based resource scaling and priority-based resource allocation | To provide autonomic resource provisioning for multilayer cloud applications | Operational costs, resource utilization, response time and profit | To solve the scaling problem of computational resources in cloud computing environment | Support only for homogenous cloud resources |
| Mazidi et al. [34] | | | | |
| Task-aware autonomic resource management Sujaudeen and Mirmalinee [35] | To effectively do the allocation when a similar task is submitted in the future by the user | Execution time, Execution cost and Resource contention | Efficiently classify the tasks based on their type, identify the relationship with the resources and allocate suitably | Self-protecting is not considered |
| | | | | |

d. **SLA Management:** The SLA management will manage the SLA among the cloud user and the provider.

e. **Workload Clustering and Scheduling:** This process clusters and schedules the specific workload to the corresponding VM based on the QoS and SLA.

### 3.1 Workload clustering

The clustering process helps to analyse the workload and describe the important features and patterns available on the cloud workload. In our work, the MDPCA is used to cluster the workload [20].

MDPCA works based on three parameters, data subset, centre selection, and similarity measurement. Initially, the centroid will be selected from the workload randomly. After that, the Euclidean distance is calculated, which follows kernel-based similarity Measure for all data points. After calculating the distance, the local density point will be grouped and create the cluster. The Gaussian kernel is used instead of the cut-off kernel to calculate the local density of the data points.

The Gaussian kernel can be expressed as follows:

$$K(\tilde{x}_i, \tilde{x}_j) = \exp \left(-\frac{||\tilde{x}_i - \tilde{x}_j||^2}{2\sigma^2}\right), \sigma > 0. \tag{1}$$

In MDPCA to compute the distance between two data points $\tilde{x}_i$ and $\tilde{x}_j$, as follows:

$$d_{i,j} = ||\phi(\tilde{x}_i) - \phi(\tilde{x}_j)|| = \sqrt{2(1 - K(\tilde{x}_i, \tilde{x}_j))}. \tag{2}$$

In Equation (1), $\tilde{x}_i, \tilde{x}_j$ represents the two data points and $K(\tilde{x}_i, \tilde{x}_j)$ denotes the kernel function of two data points, $\sigma$ which is a constant value. The mean value for every cluster will be estimated, and based on this mean value, the centroid is moved along the graph.
Algorithm 1: Workload clustering

The number of clusters needed completely will be decided by $K$.

**Step 1:** The subclasses of objects $K$ which are non-empty, will be separated randomly.

**Step 2:** At present, the centroids of clusters are the separating seed points of clusters.

**Step 3:** An object will be allotted with an object in which one has the nearer seed points.

**Step 4:** Until getting an unchangeable assignment, step 2 will be repeated.

### Table 2: Clustered workload set

| Cluster | Cluster name | QoS requirement | Resources |
|---------|-------------|-----------------|-----------|
| C1      | 334         | SLA violation, cost, availability | 4         |
| C2      | 234         | Availability, reliability, energy consumption and time | 4         |
| C3      | 138         | SLA cost, time, resource utilization | 4         |
| C4      | 463         | Execution time, energy consumption and SLA violation | 5         |

An important issue in clustering is how to determine the similarity between two objects so that clusters can be formed from objects with high similarity to each other. Euclidean distance computes the absolute differences between the coordinates of a pair of objects. A distance function yields a higher value for pairs of objects that are less similar to one another. The kernel function based distance between two items is the sum of the differences of their corresponding components.

The maximum number of the center point is taken depending upon the minimum value of the objective function. Then, the clusters are formed using those center points represented as Gang. The members in the clusters with center point surely belong to that cluster. The members other than the center point could not be assigned to any cluster, therefore, it will belong to two or more clusters. Algorithm 1 describes workload clustering process.

The clustered workload final set is given in Table 2:

#### 3.2 Workload scheduling

The workload scheduling process is conducted based on fuzzy logic [25]. Here, the resource for executing the task is served by the general expectation vector of the task. This general expectation vector is defined during the task classification. The task classification is held based on the QoS requirements and SLA agreements of users.

The general expectation vector of the task $H$ is given by,

$$ V_{wi} = (V_{wi1}, V_{wi2}, V_{wi3}) \sum_{j=1}^{3} V_{wi} = 1 $$

From Equation (3), $V_{wi1}, V_{wi2}, V_{wi3}$ represent the weights of No. of CPU, memory, and bandwidth.

Another function on a task called Justice Evaluation Function (JEF) helps to detect and prevent resource shortage from starting the task. It is calculated by the ratio of actual allotted resources and expected resources. If the justification value is lower than 1, it represents the task needs more resources to process.

The JEF for task $h$ is estimated by,

$$ J_f^i = \frac{AR}{ER} $$

where, $AR$ and $ER$ represent the allotted resource and the resource expected by the cloud user.

The actual time needed for the execution of the task is estimated by,

$$ ATE = \frac{FZ}{MIPS} $$

where, the size of the resource or the file is denoted as $FZ$ and the $MIPS$ denotes the transmission speed. (i.e. Million Information per Second).

To process the workload, the eligible VMs will be selected by comparing the $ATE$ and the time expected for the execution of the task. Additionally, the memory needed for the execution of the task and the size of the VM memory and the bandwidth needed for the execution and availability of every VMs are also compared. The justification of every class type of the task is estimated by $Jfc$ and $Jfbw$. ($Jfc$, completion time and $Jfbw$, bandwidth; both values are assumed as 0.9).

The JEF parameters are fuzzified for getting better mathematical associations which are passed through the rule base (QoS). The fuzzification is performed by using Mamdani Fuzzy inference system, and defuzzification is performed by the centroid method. It provides the output in the range of $[0, 1]$. The defuzzified result is described as $F_res$, and it is used to update the classified task. Algorithm 2 describes the task classification process.

Algorithm 2: Task classification

**Input:** Initializing tasks

**Output:** Classified tasks

**Begin**

**Step 1:** Initialize the tasks to be executed,

$$ H = h_1, h_2, \ldots, h_n $$

**Step 2:** Classify tasks based on QoS.

**Step 3:** Time needed for the execution is estimated by (23)

**Step 4:** Choose the VMs which are eligible to execute the task $h$ using GWO algorithm.

**Step 5:** Estimate the Euclidean Distance of VM.

**Step 6:** Estimate JEF using (22); $Jfc$ and $Jfbw$ are estimated by,

$$ Jfc = \sum_{i=1}^{k} \frac{ActualCt}{ExpectedCt}, Jfbw = \sum_{i=1}^{k} \frac{ActualBw}{ExpectedBw} $$

$$ Jf^i = \frac{AR}{ER} $$

$$ ATE = \frac{FZ}{MIPS} $$
where $k$ indicates the requirement No. of task

**Step 7:** Initialize the rule base and JEF parameters are given as input,

$$Rb = \{Rb_1, Rb_2, ..., Rb_k\}$$

**Step 8:** The inputted parameters are fuzzified with triangular membership function $\mu(\cdot)$.

$$\mu(\cdot) \rightarrow [0, 1]$$

$$\mu(\cdot) = \{C_i, Bw\}$$

**Step 9:** Utilize the fuzzified input parameters in rule base.

$$Rb_{if}(L, M, H) \text{and} Rb_{if}(L, M, H) \text{and} Rb_{if}(L, M, H)$$

where, $L$ indicates low, $M$ indicates medium, and $H$ indicates high.

**Step 10:** Generate $F_{\text{res}}(L, M, H) \rightarrow [0, 1]\]$$

**Step 11:** Defuzzify $F_{\text{res}}$ by centroid method.

$$F_{\text{res}} \rightarrow [F_{\text{min}}, F_{\text{max}}]$$

**End**

### 3.3 VM deployment and selection

This section presents the deployment and the selection of VMs for processing the workload with the cloud resources.

#### 3.3.1 Configuration VM

In this work, VMs are deployed randomly; the configuration of VM is shown in Section 4. Once the VMs are deployed, it then goes for the sleep state. It will wake and start to process after getting the workload from the users. These are managed by the self-management system present under the SLA; the optimal VM to process the workload is chosen by using an optimization algorithm.

#### 3.3.2 VM selection

Here, we used GWO algorithm for selecting the optimal VM based on the workload information. It follows a hierarchical style to explain the positions of grey wolves dominantly. The hierarchical patterns of the grey wolves are built by way of four levels of search agents, Alpha ($\alpha$), Beta ($\beta$), Delta ($\delta$), and Omega ($\omega$) [22]. Some decisions, such as hunting (processing), sleeping, and time to walk (awake) are made by initial level Alpha ($\alpha$). The next level Beta ($\beta$) are the subordinate wolves, which helps to decide for $\alpha$. The Delta ($\delta$) level is designed to help $\alpha$ and $\beta$. The Omega ($\omega$) is described as lower level. The first three levels ($\alpha, \beta, \delta$) are considered as the best candidate solution, it chooses the best VM based on energy, memory, and QoS needed to process the workload.

The encircling process is defined as,

$$\tilde{G} = \left| \tilde{F}_i \right| \tilde{W}_p(\text{ite}) - \tilde{W}(\text{ite}) \right|, \quad (6)$$

$$\tilde{W}(\text{ite} + 1) = \tilde{W}_p(\text{ite}) - \tilde{E}. \quad (7)$$

From Equations (6) and (7), $\tilde{W}$ indicates the location of wolf, $\tilde{W}_p$ represents the location of prey, the current iteration is represented as “ite”, and $\tilde{E}$, $\tilde{F}$ are the coefficient vectors of every iteration.

The coefficient vectors $\tilde{E}$ and $\tilde{F}$ are calculated by,

$$\tilde{E} = 2 \times \tilde{r}_1 \times \tilde{r}_1 - \tilde{m}, \quad (8)$$

$$\tilde{F} = 2 \times \tilde{r}_2. \quad (9)$$

Here, the linearly reduced components in the range of 2-0 is denoted as $\tilde{m}$. The $\tilde{r}_1$ and $\tilde{r}_2$ are the random vectors in the limit of 0 to 1.

The travelling or hunting procedure of wolf will be given by,

$$\tilde{G}_\alpha = \left| \tilde{F}_1 \tilde{W}_\alpha(\text{ite}) - \tilde{W}(\text{ite}) \right|; \tilde{G}_\beta = \left| \tilde{F}_2 \tilde{W}_\beta(\text{ite}) - \tilde{W}(\text{ite}) \right|; \tilde{G}_\delta = \left| \tilde{F}_3 \tilde{W}_\delta(\text{ite}) - \tilde{W}(\text{ite}) \right|, \quad (10)$$

$$\tilde{W}_1 = \tilde{W}_\alpha(\text{ite}) - \tilde{F}_1 \times \tilde{G}_\alpha; \tilde{W}_2 = \tilde{W}_\beta(\text{ite}) - \tilde{F}_2 \times \tilde{G}_\beta; \tilde{W}_3 = \tilde{W}_\delta(\text{ite}) - \tilde{F}_3 \times \tilde{G}_\delta, \quad (11)$$

$$\tilde{W}(\text{ite} + 1) = \frac{(\tilde{W}_1 + \tilde{W}_2 + \tilde{W}_3)}{3}. \quad (12)$$

GWO is updating the parameter $\tilde{m}$ and is linearly reorganized in every iteration to range from 2 to 0. The updating has been done through,

$$\tilde{m} = 2 - \text{ite} \cdot \frac{2}{\text{Maxite}}. \quad (13)$$

The GWO selects the VM mainly based on the QoS requirements, power needed for processing, and memory based on Algorithm 3.

### 3.4 QoS management

The QoS data provide information about the metrics used to estimate the weights for the clustered workloads. The QoS metrics used in this work are explained below:

a. **Availability:** It provides the ratio of mean time of failure (mTF) to the addition of mean time of failure and mean of
**Algorithm 3** Optimal VM selection with GWO

**Input:** Initializing randomized VMs

**Output:** Optimal VM

Initialize VM set $V_M = \{i = 1, 2, \ldots, n\}$

Initialize search agents $x, y, \text{and} \ z$

Calculate fitness function for every VM

While ($t < \text{Max number of iterations}$)

For three best VMs

Update the location of current VMs using (7)

End for

Update search agents

For all $V_M$ do

Calculate fitness function using (12)

End for

$\text{ite} = \text{ite} + 1$

End while

- **Time taken to repair** ($mTR$).
  
  \[ A = \frac{mTF}{mTF + mTR}. \]  
  \[ (14) \]

  The $mTF$ is estimated by,

  \[ mTF = \frac{\text{Total uptime}}{\text{No. of breakdowns}}. \]  
  \[ (15) \]

  The $mTR$ is estimated by,

  \[ mTR = \frac{\text{Total downtime}}{\text{No. of breakdowns}}. \]  
  \[ (16) \]

- **Reliability:** Before scheduling the resources, the reliability of every resource will be checked for fault tolerance. The reliability of the resources will be estimated by,

  \[ \text{Rel} = e^{-\lambda t}. \]  
  \[ (17) \]

  From Equation (5), $\lambda$ indicates the specific resource and indicates a number of workloads.

- **Energy consumption:** The overall utilization of energy is estimated based on the energy utilized in the processor ($p$), transceiver ($t$), memory ($m$), and in other hardware ($oh$). Which is given as,

  \[ \text{EnC} = \text{EnC}_p + \text{EnC}_t + \text{EnC}_m + \text{EnC}_oh. \]  
  \[ (19) \]

- **Execution time:** The overall time utilized for the execution is estimated by,

  \[ \text{Et} = \frac{(\text{Completion time of workload-Submission time of workload})}{n}. \]  
  \[ (20) \]

  On Equation (7), “$n$” is the number of workload.

- **SLA violation rate:** It is estimated by,

  \[ \text{SLAVR} = \text{FR} \times \sum_{i=1}^{n} w_i, \]  
  \[ (21) \]

  where, $w_i$ indicates the weight of every SLA and “FR” indicates the failure rate.

  The “FR” will be calculated by the below equation,

  \[ \text{FR} = \frac{\text{Rate of workload failure}}{\text{No. of workload}}. \]  
  \[ (22) \]

- **SLA cost:** The overall cost of SLA will be estimated by,

  \[ \text{SLAC} = \text{SLAVR} \times \sum_{i=1}^{n} \text{Execution time}, \]  
  \[ (23) \]

  where, “$i$” indicates specific resources and “$n$” indicates a number of workload.

**3.5 SLA management**

The SLA management is responsible for preparing SLA documents based on the information confirmed between the cloud client and the provider [26, 27]. These prepared documents include SLA violation information such as deviation values and penalty. With these values, the QoS deviation will be predicted. The penalty will be executed when the deviation value maximizes than permitted for workload [21].

The penalty will be provided as a free cloud service for several seconds or minutes based on the delay time. It is estimated by,

\[ \text{Penalty} = \text{Penalty}_{\text{min}} + \text{[Penalty rate} \times \text{delay time]} \]  
\[ (24) \]
Algorithm 4 Malicious detection

Input: Collected features

Output: Malicious VM

Select collected features \( z_1, z_2, z_3, \ldots, z_n \) for server \( \text{SR}_i \), \( i = 1, 2, 3, \ldots, s \)
do for VM\(_{ij} \) on \( \text{SR}_i \), \( j = 1, 2, 3, \ldots, v_i \)do
VM\(_{Mo} \) conveys the collected features \( Z_{ij} \) of VM\(_{ij} \)
Where \( Z_{ij} = \{ z_{1ij}, z_{2ij}, z_{3ij}, \ldots, z_{nij} \} \)
end
end

The rate of a penalty and minimum penalty \( s/\text{min} \) will be decided by the service provider.

3.5.1 Fault or malicious detection

The SLA management system finds any malicious attacks or faulty VM present in the network during the optimization.

a. Malicious detection: The malicious node detection process is shown in Algorithm 4. This system extracts various malicious features described in [23, 28].

The collected features \( (Z) \) is described as \( \mathbf{n} \), \( s \) is the number of servers (SR), VMs in the “SR” is denoted as \( v \). The collected features are grouped \( (Z_{ij}) \) and sent to machine learning module (mo). It includes some pre-trained modules to identify the attacks based on the malicious programs.

b. Fault detection: The anomaly VMs will be detected by fault detection scheme in SLA based on Self-organizing maps [24]. The faults are detected by comparing the performance of VMs with weight vectors and neurons present in the network. The similarity among the VMs is measured by using Euclidean distance. The threshold \( \text{th} \) will be assumed at the initial state and if the state of VMs is lower, then that VM is considered normal, and others are decided as abnormal.

Faulty(VM\(_{s} \))

\[
\begin{cases}
1, \min \left\{ \| P_v - W_{tij} \| \mid i, j = 1, 2, 3, \ldots, N \right\} \geq \text{th} \\
0, \min \left\{ \| P_v - W_{tij} \| \mid i, j = 1, 2, 3, \ldots, N \right\} < \text{th}
\end{cases}
\]

From Equation (20), VM\(_{s} \) represents the VMs in the network, each neuron weight is described as \( W_{tij} \), and the performance value of every VM is indicated as \( P_v \).

If the fault in VM is detected during the execution of any task, the task in that VM will be migrated to another non-faulty VM.

Algorithm 5 details the procedure of task migration for reducing migration times. Initially, based on the upper and lower threshold, all virtual machines are grouped. The virtual machines with the SLA rate greater than the upper threshold are added to the faulty group, and the virtual machines with the SLA rate lower than the lower threshold are added to the on-faulty group. Further, the tasks in the faulty group is selected to migrate to the non-faulty group. To minimize migration times, the tasks in the faulty virtual machine are sorted in descending order. Finally, the task will be migrated to the least SLA rate virtual machine.

Algorithm 5 Task migration

Input: Virtual machines \( \{ VM \} \), tasks \( \{ t_i \} \)

Output: Task migration

Obtaining the upper threshold \( T_{hup} \) and the lower threshold \( T_{hlow} \)
for each \( \{ VM \} \) do
if the current SLA rate \( < T_{hlow} \) then
Adding to the Non-Faulty group
else if the current SLA rate \( > T_{hup} \) then
Adding to the Faulty group
end if
end for
for faulty group do

\[ \text{taskList} \leftrightarrow \text{vm.getTaskList()} \]
\[ \text{taskList.sortDecresingbyTaskLength()} \]
end for
for non-faulty group do

\[ \text{group sorted by SLA rate in ascending order} \]
\[ \text{Migration task list sorted by execution time in descending order} \]
\[ \text{Task migrated to the low SLA rate VM} \]
end for

4 Simulation Setup and Results

The architecture of WARMS is shown in Figure 1. The simulation of WARMS technique is implemented by JAVA. The workload (Client request) used for the experimentation includes about 500–3000 requests. The results are compared based on QoS parameters such as availability, reliability, resource utilization, energy consumption, execution time, SLA violation rate, and SLA cost. The detailed description of QoS parameters are given in Section 3.4. The configuration of host and VM is shown in Tables 3 and 4.
FIGURE 1  Architecture of WARMS

TABLE 3  Host configuration used in experimentation

| S. No | Parameter        | Value     |
|-------|------------------|-----------|
| 1     | RAM (Megabytes)  | 35,000    |
| 2     | Memory (Gigabytes) | 15        |
| 3     | Energy (kilo watt) | 1000     |
| 4     | MIPS             | 2350      |
| 5     | Bandwidth (MB/s) | 1,000,000 |

TABLE 4  VM configuration used in experimentation

| S. No | Parameter        | Value     |
|-------|------------------|-----------|
| 1     | RAM (Megabytes)  | 1024      |
| 2     | Image Size (Megabytes) | 512     |
| 3     | MIPS             | 1250      |

4.1  Simulation results

The proposed WARMS has been simulated with the configuration shown in Tables 3 and 4. Initially, the performance of workload clustering and scheduling carried out by MDPC with fuzzy logic is evaluated and compared with FIFO, Fair, and K-means Clustering-based Task Classification and Scheduling based on the makespan and load standard deviation metrics. Further, the achieved performance of proposed WARMS is compared with TARNN [35], RADAR [21], and LARPA [32].

4.1.1  Makespan

Figure 2 shows that MDPC-Fuzzy techniques relatively outperforms in the 3000 workloads similar in other situations also, and other techniques achieve slight variations in all workload situations. The reasons why MDPC-Fuzzy obtains a relatively good result are the fast discovery of cluster centre and fair interpretation of workload features obtained through fuzzy rules.

4.1.2  Load standard deviation

Figure 3 displays the single index of the load standard deviation with 3000 workloads. The result illustrates that MDFC-Fuzzy has a smaller fluctuation with the load. Additionally, the load standard deviation of the other algorithms increases with some tasks, except FIFO, which shows instability.
4.1.3 Availability

Figure 4 illustrates the rate of availability based on number of workload. The WARMS system achieves varied results on a different amount of workload. It earns up to 51% with 500 workloads and it reduced when increasing the workload. Finally, with 3000 workloads, it earned 91% of the availability rate.

4.1.4 Reliability

Figure 5 shows the reliability rate with respect to number of workloads; our WARMS technique achieved up to 94% with 3000 workloads and it is much better when compared with the existing systems. The existing system earned up to 87%, 80%, and 79% with 3000 workloads. With 500 workloads, the WARMS system earned 66% reliability, which is shown in Figure 4.

4.1.5 Resource utilization

Resource utilization while processing the workload is shown in Figure 6. The WARMS techniques got a maximum resource utilization rate of 92% with 3000 workloads. Other existing systems earned 86%, 83%, and 76% on resource utilization with higher workloads. With a smaller amount of workload, both the proposed and existing systems achieved a lower utilization rate. It was caused because of the minimal amount of workload; available resources cannot be utilized by the least workloads. By increasing the workloads, the resources available on the cloud are mostly utilized completely.
4.1.6  |  Energy consumption

Energy consumption while processing the workload is shown in Figure 7. It describes that the proposed WARMS technique greatly minimized the energy consumption while comparing with other existing systems. The WARMS system consumed maximum of 420 kW of power for processing 2000 workloads, and other existing systems consumed 435, 450 and 460 kW of power for processing the workloads.

4.1.7  |  Execution time

The execution time comparison of different methods and workloads is shown in Figure 8. The execution time is measured in terms of seconds. The WARMS system utilized about 5600 to 6200 s to execute 500–3000 workloads. It is better when compared with the existing techniques.

4.1.8  |  SLA violation rate

Figure 9 illustrates the SLA violation rate based on number of workload. On the violation of SLA, our system earned lower violation rates than the existing systems. The existing techniques got a minimum violation rate of SLA of 3% to 16%. The proposed technique got a minimum of 2% on 500 workloads and a maximum of 9% on 3000 workload.

4.1.9  |  SLA cost

The SLA cost comparison of different methods and workloads are shown in Figure 10. The maximum cost utilized by the WARMS for 3000 workloads is up to $190, but the existing systems utilized $155–240 for executing 2000 workload. From the evaluation of metrics, namely availability, reliability, resource utilization, energy consumption, execution time, SLA violation
rate, and SLA cost, the proposed WARMS achieved superior results over earlier schemes. Hence, the proposed WARMS has been well suitable for autonomic resource provisioning in the cloud environment.

The ability of the GWO based VM deployment algorithm is analysed with many PMs and VMs throughout the simulation. The performance achieved is related to three VM deployment algorithms, such as ACO and PSO, and Whale optimization algorithm. Table 5 represents the results obtained based on VM allocation and migration time with different number of iteration on various techniques.

The comparative analysis is evidence that the proposed technique performed well in its respective parameters and made the cloud system as a better one. Ultimately the proposed metaheuristic technique based cloud system with optimal VM deployment and fault tolerance became a suitable option for better cloud service. The experiment is reiterated 10 times and the performances are evaluated in terms of availability, reliability, resource utilization, energy consumption, execution time, SLA violation rate, and SLA cost, which achieved better results compared to other schemes.

4.2 ANOVA test for statistical evaluation

The strength of proposed WARMS scheme is evaluated based on two significant performance metrics (resource utilization and reliability), which are relatively important for cloud resource management. To evaluate the statistical implication of the proposed scheme, we are using one-way analysis of variance (ANOVA) test. The ANOVA test has been conducted on selected parameters, which are resource utilization and reliability. Further, the outcome of hypothesis test was compared with the existing schemes, namely, TARNN, RADAR, and LARPA. In the ANOVA test, the mean value of two different schemes are taken and compared whether it is similar or not. If the samples mean value of two or more schemes are similar, then accept it for null hypothesis, and if the samples mean value of two or more schemes are dissimilar, then accept it for alternate hypothesis. The parameters employed in ANOVA test reveal the results in the form of F-statistic. The below two conditions are satisfied to reject the null hypothesis.

(i) The $p$-value should be less than the significance level.
(ii) The value of $F$-statistic must be higher than the $F$-critical value.

Also, the alternative of hypothesis $H_{null}$ can define as Equn.32 to counter the null hypothesis $H_{null}$.

$$H_{null}: \mu_{WARMS} = \mu_{TARNN} = \mu_{RADAR} = \mu_{LARPA}$$

$$H_{alt}: \mu_{WARMS} \neq \mu_{TARNN} \neq \mu_{RADAR} \neq \mu_{LARPA}$$

To perform the ANOVA test, the number of a trial, taken for evaluation is 5 for all schemes and other critical parameters such as significance level value $\alpha = 0.05$ and confidence interval (CI) = 95%. The results of the ANOVA for the schemes are shown in Tables 6 and 7. It can be observed in Tables 6 and 7 that the $F$-statistic value is greater than the $F$-critical value and simultaneously chosen level (0.05) is also greater than the
In this paper, the autonomic resource handling system called WARMS has been proposed for managing the cloud resources. The WARMS technique efficiently schedules the cloud resources automatically by the self-management system controlled under the SLA based on the QoS parameters of cloud users. Different algorithms such as MDPCA, GWO, and fuzzy are utilized to handle the self-management based system. During the experimentation of WARMS technique, we have analysed the effect of different QoS parameters including Availability, Reliability, Resource Utilization, Execution time, Energy consumption, SLA violation rate, and SLA cost with the differed amount of workload. We compared the efficiency of WARMS system with two existing cloud resource management system in terms of QoS parameters. The experimental outcome proved that the WARMS technique is better in terms of energy, cost, and time when compared with existing systems. The proposed WARMS framework successfully schedules the cloud resources automatically and maintaining the QoS and SLA during the execution of workloads, which provide more client satisfaction.

REFERENCES

1. Li, J., et al.: Secure attribute-based data sharing for resource-limited users in cloud computing. Computers & Security 72, 1–12 (2018)
2. Li, P., et al.: Privacy-preserving outsourced classification in cloud computing. Cluster Computing 1–10 (2017)
3. Stergiou, C., et al.: Secure integration of IoT and cloud computing. Future Generation Computer Systems 78, 964–975 (2018)
4. Li, Y., et al.: Intelligent cryptography approach for secure distributed big data storage in cloud computing. Information Sciences 387, 103–115 (2017)
5. Forestiero, A., et al.: Hierarchical approach for efficient workload management in geo-distributed data centers. IEEE Transactions on Green Communications and Networking 1(1), 97–111 (2017)
6. Jangtii, S., Sriram, V.S.S., Logesh, R.: The role of cloud computing infrastructure elasticity in energy efficient management of datacenters. In: 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), pp. 758–763. IEEE, Piscataway, NJ (2017)
7. Tsakalozos, K., et al.: Live vm migration under time-constraints in share-nothing iaas-clouds. IEEE Transactions on Parallel and Distributed Systems 28(8), 2285–2298 (2017)
8. Wei, W., et al.: Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing. IEEE Transactions on Services Computing 11(1), 78–89 (2018)
9. Peng, G., et al.: Knowledge-based resource allocation for collaborative simulation development in a multi-tenant cloud computing environment. IEEE Transactions on Services Computing 11(2), 306–317 (2018)
10. Yang, J., et al.: A task scheduling algorithm considering game theory designed for energy management in cloud computing. Future Generation Computer Systems (2017)
11. Liu, Y., et al.: Workload-based multi-task scheduling in cloud manufacturing Robotics and Computer-Integrated Manufacturing 45, 3–20 (2017)
34. Mazidi, A., Golsorkhtabarani, M., Tabari, M.Y.: Autonomic resource provisioning for multilayer cloud applications with K-nearest neighbour resource scaling and priority-based resource allocation. Software: Practice and Experience 50(8), 1–25 (2020)

35. Sujaudeen, N., Mimalinee, T.T.: TARNN: Task-aware autonomic resource management using neural networks in cloud environment. Concurrency and Computation: Practice and Experience e5463 (2019)

36. Zahoor, S., et al.: Cloud–fog–based smart grid model for efficient resource management. Sustainability. 10(6), 2079 (2018)

37. Xiong, Z., et al.: Cloud/fog computing resource management and pricing for blockchain networks. IEEE Internet of Things Journal 6(3), 585–600 (2018)

38. Gai, K., et al.: Resource management in sustainable cyber-physical systems using heterogeneous cloud computing. IEEE Transactions on Sustainable Computing 3(2), 60–72 (2018)

How to cite this article: Dewangan B.K. et al. Workload aware autonomic resource management scheme using grey wolf optimization in cloud environment. IET Commun.; 1–14 (2021). https://doi.org/10.1049/cmu2.12198.