Journal of Network Intelligence ©2025 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **10**, Number **2**, May **2025**

Experiment Management System Based on Software-Defined Network in Cloud Computing Environment

Miao-Miao Song*
Information Engineering College
Zhengzhou Institute of Technology, ZhengZhou 450000, P. R. China
songmiaomiao202402@163.com

Hong Fan
Colleges of Information Engineering
Centro Escolar University, Manila 1005, Philippines
dk4472@163.com

*Corresponding author: Miao-Miao Song
Received April 3, 2024, revised August 23, 2024, accepted November 27, 2024.

Abstract. *Image fuzzy enhancement is a research hotspot in the field of image processing, which aims to recover enhanced beginning clear images from degraded images. Based on the research of traditional particle swarm optimization algorithm and fuzzy enhancement algorithm, an image fuzzy enhancement method based on membrane computing particle swarm algorithm is proposed. Firstly, in order to make full use of the sparse characteristics of the clear image, the coefficient decomposition under wavelet domain and tightly supported wavelet domain is performed on the image respectively. Then, a joint optimisation model is constructed using L1 parametric constraints to achieve pretzel noise cancellation. Next, the MMH-PSO algorithm is designed by improving the particle swarm algorithm using membrane computing and Metropolis-Hastings sampling. Based on the simulated annealing algorithm temperature drop process, Metropolis-Hastings sampling is used to add randomness to the particle swarm algorithm so that it has the ability to jump out of the local optimum. The use of membrane computing enhances the parallelism of the particle swarm algorithm and can reduce the time complexity in solving complex problems. Finally, MMH-PSO is used to simultaneously search out the magnitude of the two fuzzy parameters in the traditional fuzzy enhancement algorithm in order to improve the accuracy of the algorithm. The experimental results show that the proposed algorithm has better SSIM values than the traditional fuzzy enhancement algorithm, which effectively improves the image quality and makes the image edge information more abundant.*
Keywords: Image enhancement; particle swarm algorithm; membrane computing; simulated annealing algorithm; pretzel noise

1. Introduction.

In the field of higher education, experimental teaching is an important part of cultivating students' practical ability and innovative thinking. With the rapid development of Information Technology (IT), the traditional experimental teaching mode is facing challenges such as uneven resource allocation, slow equipment updating and high maintenance cost [1, 2]. In order to improve the efficiency and quality of experimental teaching, colleges and universities are in urgent need of an experimental management system that can achieve the optimal allocation of resources, support remote access, and have the ability of flexible expansion [3].

At the present stage, experimental management systems in universities generally face problems such as inflexible resource allocation, difficulties in updating and maintenance, and low utilisation [4, 5]. As experimental equipment and resources are usually fixedly configured, it is difficult to adapt to the frequently changing needs of different courses and research projects, resulting in an unbalanced distribution of resources in time and space [6]. In addition, the slow replacement of experimental equipment makes it difficult to keep up with the pace of technological development, which affects the quality of experimental teaching and students' learning experience. In terms of maintenance, decentralised equipment management and complex network configurations increase the maintenance workload and cost [7, 8]. These problems limit the efficiency and effectiveness of experimental teaching and urgently need to be solved by technological innovation.

In this context, the convergence of cloud computing and Software-Defined Networking (SDN) technologies provides new solutions for university experiment management systems. Cloud computing provides powerful data processing and storage capabilities, making the management, analysis and sharing of large-scale experimental data more efficient. Meanwhile, SDN technology provides a more flexible and controllable network environment for experiment management systems through its flexible network resource configuration and management [9, 10]. The application of these technologies not only improves the efficiency and security of experiment management, but also promotes the optimal allocation of educational resources, accelerates the pace of educational informatisation, and provides strong technical support for the development of higher education. Cloud computing provides elastic resource management and on-demand services, while SDN enables dynamic adjustment of network traffic and resources through centralised control and network programmability. The introduction of such systems not only enhances the speed of setting up experimental environments and the utilisation of experimental resources, but also meets the needs of different experimental courses through flexible network configuration, while reducing the complexity and cost of experimental management.

1.1. **Related work.** Current research in the area of dynamic resource management for cloud computing platforms typically focuses on optimising the allocation and utilisation of computing resources in cloud environments. The main areas of interest include: resource allocation algorithms [11], scalability [12], performance optimisation [13], energy efficiency [14], virtualisation [15] and reliability [16].

Islam et al. [17] proposed an empirical prediction model based on neural networks and linear regression for adaptive resource allocation in cloud environments. The model aims to provide adaptive resource management for cloud hosted applications. It was found that although the approach improves the adaptability of resource allocation, the latency introduced in hardware resource allocation and the accuracy of the model under different conditions need to be further explored. Addis et al. [18] developed a hierarchical approach for resource management of very large cloud platforms, which was designed based on mixed integer nonlinear optimisation. By dynamically adapting cloud resource allocation to meet performance requirements and minimise energy costs, the study highlights the complexity of dynamic resource management in large-scale cloud service centres. García et al. [19] investigated the dynamic resource management problem in cloud computing by improving resource utilisation through improved allocation and migration of virtual machines and server consolidation strategies. The study revealed the challenges in achieving effective server consolidation and VM allocation to reduce carbon emissions and improve resource utilisation. Mazumdar and Pranzo [20] focused on dynamic resource management of CPU and memory in cloud data centres using specific algorithms to conserve underutilised resources. The paper states that finding the most efficient algorithms to prevent underutilisation and wastage of resources is a key challenge in this area. Wu et al. [21] proposed an efficient dynamic system resource management approach to handle dynamic network loads in cloud computing environments through an optimal load balancing controller. It is shown that the method has advantages in terms of control accuracy and stability, but the ability to manage system noise under different workload modes needs further verification. Alzhouri et al. [22] explored a dynamic resource management model in the cloud point-of-sale market with the aim of expanding cloud revenues by managing excess resources. The challenge of this research is to optimise cloud resources when responding to random customer demands and strategies. Aldossary [23] conducted a comprehensive literature review focusing on dynamic resource management in cloud computing environments including VM consolidation and resource provisioning techniques. The study highlighted the need to consider the additional costs of migration, scaling time and energy overheads to balance the limitations of these techniques. Swain et al. [24] conducted an extensive survey on resource management schemes in cloud environments, especially machine learning based resource allocation strategies. The paper discusses various challenges such as processing/overload handling, resource wastage and VM migration and points out the importance of determining effective Virtual Machine Placement (VMP) to minimise resource wastage and operational costs.

1.2. Motivation and contribution.

In traditional IT lab management systems, network configuration and management are usually performed manually, which is not only time-consuming but also error-prone. A policy-driven SDN system automates network configuration and management through a centralised control platform, making the deployment and modification of network policies more flexible and efficient. In addition, resource allocation in traditional IT lab management systems tends to be static, which means that resources may be over-allocated to cope with peak loads, resulting in low resource utilisation during off-peak hours. To address the above issues, this paper designs a policy-driven SDN system and proposes an auto-scaling algorithm for dynamic management of cloud resources. The main innovations and contributions of this work include:

(1) Proposes a policy definition that employs a ternary structure based on Subject, Action and Condition, providing clear semantics and easy implementation. Resource allocation policies follow this structure to enable dynamic allocation and optimisation of computing, storage and network resources. Policy prioritisation and conflict resolution mechanisms ensure consistency and predictability of system decisions.

(2) A system processing flow algorithm is proposed that can receive experiment task requests submitted by users, dynamically allocate resources and configure the SDN network topology. By monitoring resource usage and dynamically adjusting resource allocation, it ensures the smooth execution of experimental tasks.

(3) For the load characteristics of virtual network labs, an improved prediction algorithm combining wavelet transform and Wavelet-MLE is proposed to effectively capture the self-similarity and suddenness characteristics of the load. An auto-scaling algorithm based on the prediction results automatically adjusts VM resources to meet the load demand and optimise resource utilisation. The cost model evaluates the resource utilisation, quality of service and operation cost for cost optimisation.

2. Concepts and technologies related to IT laboratory management systems.

2.1. Cloud computing technology.

Cloud computing, as an emerging information technology service model, is centred on the provision of dynamically allocable and scalable computing resources and services through the network. The IT laboratory management system in the cloud computing environment can make full use of the elasticity and scalability of the cloud to provide more flexible and efficient support for experiment management. The service model of cloud computing mainly includes three kinds: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In the IT lab management system, the appropriate service model can be selected according to the actual needs in order to achieve the optimal allocation and management of resources.

Cloud computing deployment models include private, public, hybrid and community clouds. IT lab management systems can adopt private or hybrid cloud models to ensure data security and resource exclusivity, while also taking advantage of the elasticity and shared nature of public clouds.

2.2. **IT laboratory management system.** IT lab management system is a comprehensive information system applied in IT lab environment for coordinating and managing lab resources, experimental processes, user rights and so on. The system aims to improve the utilisation of laboratory resources, simplify the experimental process and guarantee the security of experimental data.

The main objectives of IT lab management system include: improving the efficiency of lab resources utilisation, simplifying the lab management process, enhancing the quality of experimental teaching, and guaranteeing the stability and security of the experimental environment. The main functions of IT lab management system include: lab resource management, experimental process management, user rights management, experimental data management, network traffic monitoring, etc. We can dynamically allocate computing resources according to user demand, as shown below:

$$R = f(C, U, T) \tag{1}$$

where R denotes the amount of resources allocated, C denotes the resources requested by the user, U denotes the user type, and T denotes the time.

Minimising cost while meeting quality of service (QoS) requirements [25].

$$Cost = \alpha \cdot \left(Resource_{Usage} + Overhead\right) \tag{2}$$

where α is the cost factor, $Resource_{Usage}$ is the resource usage, and $Overhead$ is the management overhead.

In addition, we need to ensure that the performance of the service meets the QoS requirements of the users.

$$QoS = f(Latency, Throughput, Availability) \tag{3}$$

where $Latency$ denotes latency, $Throughput$ denotes throughput, and $Availability$ denotes availability.

Efficiently migrate and replicate data in a cloud environment to ensure data consistency and reliability.

$$ReplicationFactor = ceil(W/T) \tag{4}$$

where W is the write load and T is the replica threshold.

2.3. **Introduction to Software Defined Networking.** SDN is an innovation in network architecture, the core idea of which is to control the behaviour of network hardware through software applications, thus achieving dynamic programmability of the network. In IT lab management systems, SDN technology can provide more flexible network configuration and management to adapt to changing experimental needs.

The SDN architecture consists of three main components: the SDN controller, the southbound interface, and the northbound interface. The SDN controller is responsible for issuing network configuration commands, the southbound interface is used for communication between the controller and the network hardware, and the northbound interface is provided for network applications and services. In IT lab management systems, SDN technology can be applied to scenarios such as the creation of virtual lab environments, the monitoring and control of network traffic, and the dynamic allocation of experimental resources. SDN provides multiple advantages in cloud-based IT lab management systems, which are mainly reflected in the following aspects:

(1) Centralised management: SDN allows network administrators to manage the entire network through a centralised controller, rather than dispersing it across individual physical devices. This centralised management simplifies network configuration and monitoring, making it more efficient to manage large amounts of virtual network resources in a cloud environment.

(2) Flexibility and programmability: The programmability of SDN allows network administrators to dynamically adjust network configurations by writing scripts to adapt to changing cloud resources and user requirements. This flexibility means that the network can respond quickly to changes in computing resources, such as the startup and migration of virtual machines.

(3) Policy-driven network control: With a policy-driven approach, SDN allows administrators to define high-level network policies without concern for the underlying implementation details. This enables users who are not network experts to control network behaviour through policies, improving the availability and security of lab resources.

(4) Resource optimisation: SDN helps to optimise the use of network resources by dynamically allocating network bandwidth and routing policies to accommodate different workload demands. This improves resource utilisation and reduces resource wastage in cloud computing environments.

In summary, SDN brings higher management efficiency, flexibility, security and resource utilisation to cloud-based IT lab management systems, helping to build a more robust and reliable network environment.

2.4. **Auto-scaling resource allocation technology.** Virtualisation technology, which is the foundation of cloud computing and SDN technology, allows multiple virtual machines to run on a single physical server, providing IT lab management systems with the ability to isolate and dynamically allocate resources. Automation technology enables IT lab management systems to automate many management tasks, such as environment deployment, resource allocation, and performance monitoring, which greatly improves management efficiency and accuracy.

Auto-Scaling (AS) resource allocation technique is a technique that dynamically adjusts resources based on the actual workload of an application [26, 27]. This technique is particularly important in cloud computing environments as it helps systems to maintain performance and cost efficiency in the face of changing loads. The auto-scaling principle consists of five main related computations as shown below:

(1) Load balancing calculation

$$L(t) = \frac{C(t)}{N(t)} \tag{5}$$

where $L(t)$ is the average load at time t; $C(t)$ is the total computational demand (workload) at time t; and $N(t)$ is the number of resource instances at time t. This method is used to determine the average load that each resource instance should carry at a given point in time.

(2) Calculation of resource requirements

$$R(t) = \frac{C(t)}{T_{\max}} \tag{6}$$

where $R(t)$ is the resource requirement at time t; $C(t)$ is the total computational requirement at time t; and $T_{\max}$ is the maximum load that a single resource instance can handle. This method is used to calculate the number of resources the system needs without exceeding the maximum load of a single instance.

(3) Automatic Scaling Decision Calculation

$$N_{\text{new}} = N_{\text{current}} + \text{round}\big(R(t) - N_{\text{current}}\big) \tag{7}$$

where N_{new} is the adjusted number of resource instances; N_{current} is the current number of resource instances; and $R(t)$ is the resource demand at time t. This method is used to decide the number of resource instances to add or reduce based on the current demand and existing resources.

(4) Calculation of the rate of change of load within the time window

$$LoadChangeRate = \frac{L(t_2) - L(t_1)}{t_2 - t_1} \tag{8}$$

where $L(t_1)$ and $L(t_2)$ are the loads at time t_1 and t_2, respectively; and $t_2 - t_1$ is the length of the time window. This method is used to measure the rate of change of the load within a given time window to facilitate scaling decisions.

(5) Cost-benefit calculations

$$Cost\_Benefit = \frac{Performance\_Gain}{Cost\_Increase} \tag{9}$$

This equation is used to evaluate whether the performance improvement from adding resources outweighs the increase in additional cost. If $Cost\_Benefit > 1$, scaling is considered beneficial; if $Cost\_Benefit < 1$, other optimisation strategies may need to be considered.

The key to auto-scaling resource allocation techniques is the ability to accurately assess the real-time load on the system and dynamically adjust resources based on the load to keep the system running efficiently and cost-effectively. With the above equation, IT lab management system administrators can better implement auto-scaling strategies.

3. **Policy-driven SDN system design.**

3.1. **Policy definition.**

At the heart of software-defined networking (SDN)-based experiment management systems in cloud computing environments is policy definition [28, 29]. Policy definitions allow users and administrators to describe network behaviours and resource management rules in a high-level manner without delving into the underlying network configuration details. In this study, policy definitions are the core mechanism for implementing SDN control logic, which allows the system to express network behaviour and resource management rules in a declarative manner. Policy definitions not only simplify network management, but also improve network flexibility and programmability.

The SDN controller converts policies into specific network configuration and routing decisions to achieve effective management of network resources in the experiment management system. The definition and management of policies is the key to achieving efficient, reliable and secure experiment management in cloud computing environments. A policy consists of three main parts: Subject, Action and Condition. The Subject usually refers to a user or application in the network, the Action defines the operations that can be performed by the Subject, such as reading, writing, or communicating, and the Condition describes the specific environment or state that needs to be satisfied for the action to be executed. This ternary structure makes policies both semantically clear and easy to implement and verify. The generic structure of a policy description is shown below:

$$Policy(p) = \{Subject(s), Action(a), Condition(c)\} \tag{10}$$

where $Policy(p)$ denotes a specific policy, $Subject(s)$ denotes the subject of the policy (e.g., a user or an application), $Action(a)$ denotes an action that is allowed or denied by the policy, and $Condition(c)$ denotes a condition that needs to be met before the action is executed. Equation (10) provides a common structure for network policies, making policy creation and management more intuitive.

Resource allocation policies are defined based on the policy description Equation (10), which specifies which resources should be allocated to specific users or tasks. By combining policies with resources, the system is able to achieve dynamic allocation and optimisation of computing, storage and network resources.

$$ResourcePolicy(r) = Policy(p) \rightarrow Resource(res) \tag{11}$$

where $ResourcePolicy(r)$ denotes a resource-related policy and $Resource(res)$ is a specific network resource (e.g., bandwidth, number of connections, etc.). Combine generic policies with specific resources to achieve fine-grained control over resources.

Since there may be multiple concurrently applicable policies, a mechanism is needed to resolve potential conflicts. The policy priority equation provides a way to specify the priority order of policies. Higher priority policies will override lower priority policies, ensuring consistency and predictability in system decision making.

$$Priority(pr) = Policy(p) \rightarrow PriorityValue(val) \tag{12}$$

where $Priority(pr)$ denotes the priority of the policy, $Policy(p)$ is the specific policy, and $PriorityValue(val)$ is a numeric value used to decide which policy is to be executed first when multiple policies are applicable at the same time. A policy with a high priority will override a policy with a low priority.

When two or more strategies conflict with each other, a conflict resolution equation is used to determine which strategy should be executed. This is usually based on the priority of the strategy, but other factors such as the age or origin of the strategy may also be considered. Equation (13) is used to resolve strategy conflicts when two or more strategies apply under the same conditions.

$$ConflictResolution(Policy_1, Policy_2) = \mathrm{Max}\big(Priority(pr_1), Priority(pr_2)\big) \tag{13}$$

where $Policy_1$ and $Policy_2$ are potentially conflicting policies; $Priority(pr_1)$ and $Priority(pr_2)$ are their priorities, respectively. By comparing the priorities and selecting the higher one, the system is able to execute the policies consistently.

In order to ensure that the policies are effective in achieving the desired goals, the effectiveness of the policy execution needs to be evaluated. The evaluation equation allows administrators to quantify the effectiveness of a policy and adjust the policy accordingly to improve the overall performance and resource utilisation of the system.

$$Effectiveness(e) = \frac{AchievedGoals(g)}{TotalGoals(tg)} \times Timeliness(t) \tag{14}$$

where $Effectiveness(e)$ denotes the effectiveness of the strategy execution, $AchievedGoals(g)$ is the number of goals reached by the strategy execution, $TotalGoals(tg)$ is the total number of goals set, and $Timeliness(t)$ is the timeliness indicator of the strategy execution. Equation (14) helps the administrator to evaluate whether the policy achieves the expected results and adjust the policy in time to improve the management efficiency.

With the above policy definitions, the SDN system can flexibly adjust network resources according to user demands and system status to achieve efficient experiment management and optimisation. In practice, the policy definition needs to be tightly integrated with the SDN controller so that it can respond to environmental changes and user demands in real time.

3.2. System processing flow algorithm.

The main function of the proposed algorithm is to manage the SDN-based experimental resources in the cloud computing environment. It can dynamically deploy computing, storage and network resources and configure the SDN network topology according to the demands of the experiment tasks submitted by users to ensure that the experiment tasks can be executed smoothly.

Specifically, the workflow of the proposed algorithm is as follows: firstly, it receives the experimental task request submitted by the user, and analyses the resource demand required for the experiment. Then, query the available resources under the management of the current SDN controller, and match the resource requirements with the available resources. If the available resources can satisfy the experimental requirements, the resources are directly allocated and the SDN network topology is configured; if the resources are insufficient, the dynamic scaling mechanism is triggered to increase the resources and re-allocate them. During the execution of the experiment, the algorithm also continuously monitors the resource usage and dynamically adjusts the resource allocation. At the end of the experiment, all previously allocated resources are released.

The pseudo-code of the processing flow algorithm of the system proposed in this work is shown in Algorithm 1.

Algorithm 1 System processing flow

```
Input: Experiment task request
Output: Experiment resource allocation plan
 1: Get the experiment topology
 2: Get the requirements for computing, storage, and network resources
 3: Query the SDN controller status
 4: Get the available computing, storage, and network resources
 5: Match the resource requirements with the available resources
 6: if the resources meet the experiment requirements then
 7:     Allocate the computing, storage, and network resources
 8:     Configure the SDN network topology
 9:     return the experiment resource allocation plan
10: else
11:     Trigger the dynamic scaling mechanism
12:     for each resource type do
13:         Call the scaling-up algorithm to increase the resources
14:     end for
15:     Re-allocate the computing, storage, and network resources
16:     return the experiment resource allocation plan
17: end if
18: Monitor the experiment running status
19: Dynamically adjust the resource allocation
20: Release the resources after the experiment ends
```

As shown in Figure 1, there is a network management module in the cloud manager. When the cloud manager needs to create or make use of network resources, it needs to call the network management module. The network management module then invokes external SDN controllers through the SDN plug-ins in it [30]. Since there are multiple SDN controllers (e.g., Floodlight, POX, Ryu, etc.), different SDN controllers are invoked by installing different plug-ins in the network management module. In this case, the invocation is typically done through a RESTful API, where the SDN plugin sends an HTTP request.

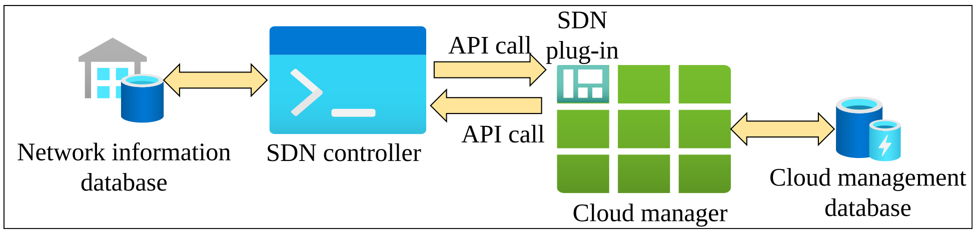

Figure 1. API Calls between Cloud Database and SDNs

4. Algorithms for dynamic management and auto-scaling of cloud resources.

4.1. Load Prediction in Cloud Environments.

In cloud computing environments, dynamic allocation and scheduling of resources requires predicting future system loads to determine whether resources need to be expanded or contracted. Load prediction is a key part of the auto-scaling algorithm. In this paper, we analyse the self-similarity of load patterns for the characteristics of virtual network labs and propose an improved prediction algorithm.

4.1.1. *Lab load characterisation.* Users of the virtual network lab generally request services in an obvious bursty and cyclical manner. As shown in Figure 2, we monitored the actual load of the virtual network lab system for a week and found that its load pattern has the following characteristics.

(1) *Burst load.* During certain specific time periods of the day, there will be a dramatic increase in load, such as the morning peak from 8:00–10:00 and the evening peak from 19:00–21:00. These peaks correspond to the times when students are attending labs and self-study sessions.

(2) *Periodicity pattern.* The daily load curve is basically repeated, showing obvious periodic characteristics. Meanwhile, on weekends, the load will also drop significantly.

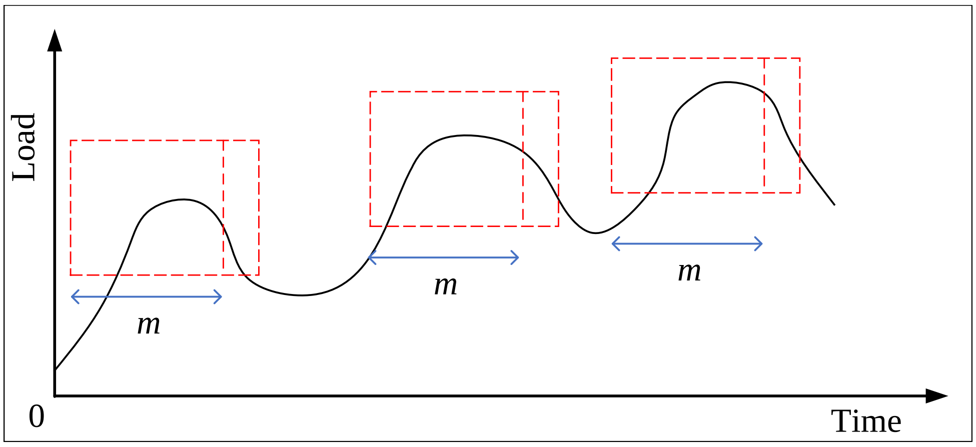

Figure 2. API Calls between Cloud Database and SDNs

4.1.2. *Laboratory load prediction method.* In order to be able to effectively predict virtual network laboratory loads with self-similarity and burstiness, this paper proposes an improved prediction algorithm that incorporates self-similarity modelling based on wavelet transform and burstiness features of wavelet maximum likelihood estimation (Wavelet-MLE). The Wavelet-MLE algorithm flow is as follows.

Step 1: Decompose the original load data sequence by wavelet transform, and get the approximate sequence and detailed sequence with different scales.

Step 2: Use maximum likelihood estimation to estimate the self-similarity parameter H corresponding to each detail sequence, and build the fractional Gaussian noise model.

Step 3: According to the self-similarity parameter H estimated for each detail sequence, wavelet maximum likelihood prediction is performed on the detail sequences.

Step 4: The predicted detail sequence and approximate sequence are wavelet reconstructed to get the predicted value of future load.

The improved algorithm is able to capture the suddenness and self-similarity characteristics of the load sequence well, and the prediction performance is better than the pure wavelet prediction or ARIMA and other statistical models. The specific prediction methods are as follows:

Let the original load sequence be X, and perform wavelet decomposition to obtain an approximate sequence A_j with scale j and a detail sequence D_j: the

$$X = A_J + \sum_{j=1}^{J} D_j \tag{15}$$

For each detail sequence D_j, estimate its self-similarity parameter H_j, and build the following fractional Gaussian noise model.

$$D_j(k) = \frac{1}{m_j} \sum_{i=1}^{m_j} \alpha_{j,i}\, \phi\left(\frac{k - k_{j,i}}{m_j^{H_j}}\right) \tag{16}$$

where $\phi(x)$ is the normalised fractional Gaussian noise function. After estimating each $\alpha_{j,i}$ and $k_{j,i}$ using the Wavelet-MLE method, the future values of the detail series can be predicted.

Finally, the predicted detail sequence is wavelet reconstructed with the approximate sequence A_J to obtain the predicted values of future loads.

$$\hat{X}(n) \ = \ \hat{A}_J(n) \ + \ \sum_{j=1}^{J} \hat{D}_j(n) \tag{17}$$

By predicting the future load, it provides a basis for dynamic scheduling and auto-scaling of resources. The design of the auto-scaling algorithm based on load prediction will be introduced next.

4.2. **Auto-scaling algorithm based on load prediction.** Based on the predicted future load scenarios, this paper proposes an auto-scaling algorithm for automatically adjusting VM resources to meet the load demand and optimise resource utilisation. The goal of the auto-scaling algorithm is to minimise the cost while meeting the quality-of-service requirements. A key objective in designing an auto-scaling strategy is to optimise cost, which includes but is not limited to reducing unnecessary resource allocation and eliminating resource wastage. To achieve cost optimisation, a cost model is used to evaluate the economic benefits of different scaling strategies. The cost model is evaluated based on the following aspects: (1) Resource utilisation: improving resource utilisation reduces idle resources and thus reduces the overall cost. (2) Quality of Service: Ensure that the quality of service is within the acceptable range to avoid unavailability of service or long response time due to scaling operation. (3) Operation cost: The scaling operation itself also incurs costs, including the creation, destruction and migration of virtual machines.

Firstly, the initial load is empirically predicted and VM clusters are started. Then, the number of resource requests, the number of VM clusters, the number of running services, and the average utilisation of the VM clusters are collected. Future load requests are predicted using a load predictor. Next, the number of VM instances is adjusted based on the prediction results and current resource status. Finally, cost optimisation is performed, evaluating all possible resource adjustment options and selecting the lowest cost option for implementation.

The objective of cost optimisation is to find the optimal allocation of resources to reduce operating costs. The cost function C can be expressed as:

$$C_{\text{total}} = C_{\text{resource}} + C_{\text{service}} + C_{\text{operation}} \tag{18}$$

where C_{total} denotes the total cost; C_{resource} represents the resource cost, which is related to the resource utilisation rate and the resource price; C_{service} denotes the quality of service cost, which is evaluated based on the time of unavailability of the service and the scope of the impact; $C_{\text{operation}}$ is the operation cost, which relates to the execution cost of scaling operations.

Auto-scaling algorithms based on load prediction consider self-similarity and time series properties of loads. Self-similarity refers to the fact that time series have similar statistical properties on different time scales. By analysing the historical load data, we can discover the self-similar characteristics of the load and use these characteristics to build a prediction model. The quantitative description of self-similarity is shown below:

$$H(q) = \tfrac{1}{2}\log\left(\frac{S(t)}{S(2t)}\right) \tag{19}$$

where $H(q)$ is the Hurst exponent, q is the time scale, and $S(t)$ is the standardised coefficient of variation at time scale t. Hurst exponents close to 0.5 indicate that the time series are self-similar.

The auto-scaling algorithm uses the Wavelet-MLE algorithm to predict future loads and dynamically adjusts resources based on the predictions. The core of the auto-scaling algorithm is to minimise the total cost of ownership while guaranteeing the quality of service.

Suppose $L(t)$ denotes the system load at time t and $L_{\text{predict}}(t + \Delta t)$ denotes the predicted future load. When $L_{\text{predict}}(t + \Delta t)$ exceeds the threshold T_{up}, a scaling operation is performed; when $L_{\text{predict}}(t + \Delta t)$ is below the threshold T_{down}, a scaling operation is performed. The decision of the scaling operation can be expressed as follows.

Expansion: If $L_{\text{predict}}(t + \Delta t) > T_{\text{up}}$, then add N resource units.

Reduction: If $L_{\text{predict}}(t + \Delta t) < T_{\text{down}}$, then reduce by N resource units.

Here, N is determined based on the cost optimisation model and the quality-of-service requirements to ensure that the cost is reduced as much as possible while satisfying the quality of service. With the above approach, an auto-scaling strategy that can both predict future load changes and automatically adjust resources based on the cost optimisation model can be implemented to effectively manage the dynamic resources on the cloud computing platform.

5. **Experimental results and analyses.**

5.1. **Experimental setup.** The experiments are conducted through the Openstack cloud platform. In the neutron network management module of Openstack, external SDN controllers can be supported through plugins. Since the proposed SDN system is developed based on Floodlight, it is also recognised by the Openstack Floodlight plugin and can be well integrated with the Openstack platform [31, 32]. The version of Openstack used for the experiments is the Havana version, deployed on top of three nodes.

The server hardware configuration for each node is DELL PowerEdge R720, CPU: Intel-Xeon E5-2650, 32GB RAM. The operating system of each server is Ubuntu 12.04 LTS 64 bit. The roles of the three servers are as follows: the first one is used as a Cloud Controller and some Openstack core services; the second one is used as a neutron network management node and a compute node; and the third one is dedicated as a compute node

5.2. **The impact of SDN on multiple users.** Figure 3 illustrates the amount of load increase for the two types of CPUs in the system as the number of users increases. The horizontal coordinate represents the number of users in either actual number or percentage. The vertical coordinate represents the increase of CPU load in percentage (%).

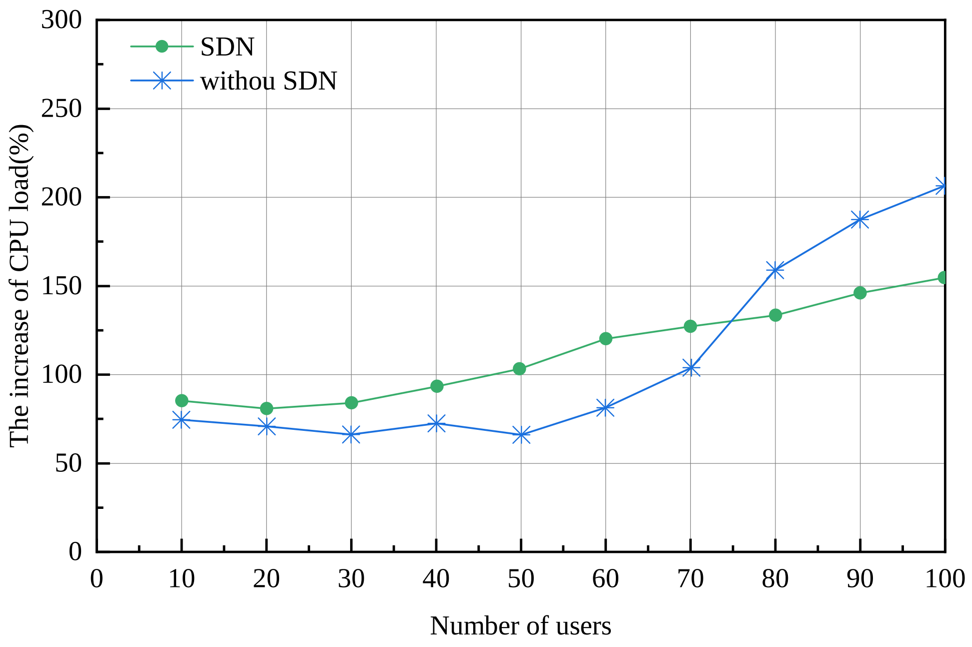

Figure 3. Increase in system load when increasing the number of users

The rapid increase in CPU load with the increase in the number of users indicates that it is closely related to the number of users and requires additional resources with the increase in users. The relatively smooth increase in CPU load for the SDN system compared to the system without SDN suggests that the increase in the number of users has less of an impact on these components. Specifically, SDN services require less resource investment as the number of users increases, which means that resource planning and expansion plans do not need to focus on services that are closely related to the increase in the number of users.

5.3. **The impact of Auto-scaling.** Figure 4 shows the three methods for accumulating the total cost at different time intervals. The results show that horizontal scaling is the worst method which shows 13% more cost than the auto-scaling method. Lightweight scaling and auto-scaling of virtual resources take different cost considerations at different levels, so that both methods consume relatively similar resources and costs most of the time. Even so, the cost of the method in this paper is 3% lower than the lightweight scaling method.

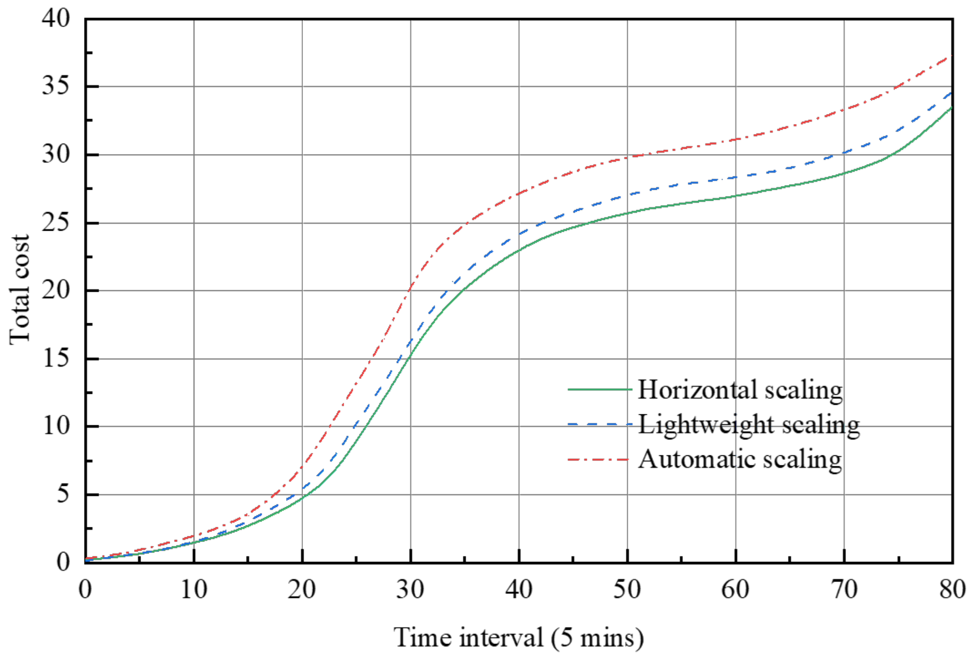

Figure 4. Total costs at different time intervals

6. **Conclusion.** This work designs a policy-driven SDN system and proposes an auto-scaling algorithm for dynamic management of cloud resources. Firstly, a policy definition is proposed that employs a ternary structure based on Subject, Action and Condition. The resource allocation policy follows this structure to achieve dynamic allocation and optimisation of computing, storage and network resources. Second, a system processing flow algorithm is proposed that can receive experiment task requests submitted by users, dynamically allocate resources and configure the SDN network topology. Finally, for the load characteristics of virtual network labs, an improved prediction algorithm combining wavelet transform and Wavelet-MLE is proposed to effectively capture the self-similarity and suddenness characteristics of the load. The auto-scaling algorithm based on the prediction results automatically adjusts the VM resources to meet the load demand and optimise the resource utilisation. Experimental results show that SDN services require less resource investment as the number of users increases. The cost of the auto-scaling algorithm is also 3% lower than the lightweight scaling algorithm.

REFERENCES

[1] S. H. Thomke, "Managing experimentation in the design of new products," *Management Science*, vol. 44, no. 6, pp. 743–762, 1998.

[2] R. Croson and K. Donohue, "Experimental economics and supply-chain management," *Interfaces*, vol. 32, no. 5, pp. 74–82, 2002.

[3] W. J. Evers, A. Brouwers, and W. Tomic, "A quasi-experimental study on management coaching effectiveness," *Consulting Psychology Journal: Practice and Research*, vol. 58, no. 3, pp. 174, 2006.

[4] C. Allan, J.-M. Burel, J. Moore, C. Blackburn, M. Linkert, S. Loynton, D. MacDonald, W. J. Moore, C. Neves, and A. Patterson, "OMERO: flexible, model-driven data management for experimental biology," *Nature Methods*, vol. 9, no. 3, pp. 245–253, 2012.

[5] J. C. Castilla, "Roles of experimental marine ecology in coastal management and conservation," *Journal of experimental marine Biology and Ecology*, vol. 250, no. 1-2, pp. 3–21, 2000.

[6] T.-Y. Wu, F. Kong, Q. Meng, S. Kumari, and C.-M. Chen, "Rotating behind security: an enhanced authentication protocol for IoT-enabled devices in distributed cloud computing architecture," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, 36, 2023.

[7] T.-Y. Wu, L. Wang, X. Guo, Y.-C. Chen, and S.-C. Chu, "SAKAP: SGX-Based Authentication Key Agreement Protocol in IoT-Enabled Cloud Computing," *Sustainability*, vol. 14, no. 17, 11054, 2022.

[8] T.-Y. Wu, Q. Meng, S. Kumari, and P. Zhang, "Rotating behind Security: A Lightweight Authentication Protocol Based on IoT-Enabled Cloud Computing Environments," *Sensors*, vol. 22, no. 10, 3858, 2022.

[9] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, pp. 102595, 2020.

[10] A. Shirmarz and A. Ghaffari, "Performance issues and solutions in SDN-based data center: a survey," *The Journal of Supercomputing*, vol. 76, no. 10, pp. 7545–7593, 2020.

[11] L. M. Haji, S. Zeebaree, O. M. Ahmed, A. B. Sallow, K. Jacksi, and R. R. Zeabri, "Dynamic resource allocation for distributed systems and cloud computing," *TEST Engineering & Management*, vol. 83, pp. 22417–22426, 2020.

[12] J. Praveenchandar and A. Tamilarasi, "RETRACTED ARTICLE: Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 4147–4159, 2021.

[13] R. K. Naha, S. Garg, A. Chan, and S. K. Battula, "Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment," *Future Generation Computer Systems*, vol. 104, pp. 131–141, 2020.

[14] T. Khan, W. Tian, G. Zhou, S. Ilager, M. Gong, and R. Buyya, "Machine learning (ML)-centric resource management in cloud computing: A review and future directions," *Journal of Network and Computer Applications*, vol. 204, pp. 103405, 2022.

[15] A. Jyoti and M. Shrimali, "Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing," *Cluster Computing*, vol. 23, no. 1, pp. 377–395, 2020.

[16] C. Li, J. Bai, Y. Chen, and Y. Luo, "Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system," *Information Sciences*, vol. 516, pp. 33–55, 2020.

[17] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[18] B. Addis, D. Ardagna, B. Panicucci, M. S. Squillante, and L. Zhang, "A Hierarchical Approach for the Resource Management of Very Large Cloud Platforms," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 253–272, 2013.

[19] A. G. García, I. B. Espert, and V. H. García, "SLA-driven dynamic cloud resource management," *Future Generation Computer Systems*, vol. 31, pp. 1–11, 2014.

[20] S. Mazumdar and M. Pranzo, "Power efficient server consolidation for cloud data center," *Future Generation Computer Systems*, vol. 70, pp. 4–16, 2017.

[21] T. Wu, S. Wang, and X. Shi, "Efficient dynamical system resource management method in cloud computing," *The Journal of Engineering*, vol. 2019, no. 23, pp. 8891–8894, 2019.

[22] F. Alzhouri, S. B. Melhem, A. Agarwal, M. Daraghmeh, Y. Liu, and S. Younis, "Dynamic Resource Management for Cloud Spot Markets," *IEEE Access*, vol. 8, pp. 122838–122847, 2020.

[23] M. Aldossary, "A Review of Dynamic Resource Management in Cloud Computing Environments," *Computer Systems Science and Engineering*, vol. 36, no. 3, pp. 461–476, 2021.

[24] S. R. Swain, A. Parashar, A. K. Singh, and C. N. Lee, "Efficient straggler task management in cloud environment using stochastic gradient descent with momentum learning-driven neural networks," *Cluster Computing*, vol. 12, no. 4, pp. 461–476, 2023.

[25] P. Goyal, V. Rishiwal, and A. Negi, "A comprehensive survey on QoS for video transmission in heterogeneous mobile ad hoc network," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 7, e4775, 2023.

[26] S. Chouliaras and S. Sotiriadis, "An adaptive auto-scaling framework for cloud resource provisioning," *Future Generation Computer Systems*, vol. 148, pp. 173–183, 2023.

[27] G. Sheganaku, S. Schulte, P. Waibel, and I. Weber, "Cost-efficient auto-scaling of container-based elastic processes," *Future Generation Computer Systems*, vol. 138, pp. 296–312, 2023.

[28] M. S. Farooq, S. Riaz, and A. Alvi, "Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review," *Electronics*, vol. 12, no. 14, 3077, 2023.

[29] L. M. Halman and M. J. Alenazi, "MCAD: A Machine learning based cyberattacks detector in Software-Defined Networking (SDN) for healthcare systems," *IEEE Access*, vol. 11, pp. 37052–37067, 2023.

[30] K. Rui, H. Pan, and S. Shu, "Secure routing in the Internet of Things (IoT) with intrusion detection capability based on software-defined networking (SDN) and Machine Learning techniques," *Scientific Reports*, vol. 13, no. 1, 18003, 2023.

[31] S. Faezi and A. Shirmarz, "A comprehensive survey on machine learning using in software defined networks (SDN)," *Human-Centric Intelligent Systems*, vol. 3, no. 3, pp. 312–343, 2023.

[32] A. H. Alomari, S. K. Subramaniam, N. Samian, R. Latip, and Z. Zukarnain, "Dual-Phase Resource Allocation Algorithm in Software Defined Network SDN-Enabled Cloud," *IEEE Access*, vol. 11, pp. 102301–102315, 2023.