International Journal for Housing Science and Its Applications
Publish August 3, 2025. Volume 46, Issue 3 Pages 495-506

https://doi.org/10.70517/ijhsa46335

Research on the Application of Computational Modeling in Unified Management Techniques for Multi-Cloud Heterogeneous Resources

Zhengxiong Mao[1,*], Yan Shi[2] and Yonghui Ren[3]
[1] Information Center of Yunnan Power Grid Co., Ltd., Chongqing University, Kunming, Yunnan, 650000, China
[2] Information Center of Yunnan Power Grid Co., Ltd., Yunnan Normal University, Kunming, Yunnan, 650000, China
[3] Information Center of Yunnan Power Grid Co., Ltd., Yunnan University, Kunming, Yunnan, 650000, China
Corresponding authors: (e-mail: mzx998yunnan@163.com).

Abstract With the rapid development of cloud computing technology, unified management of multi-cloud heterogeneous resources has become a key technology to improve resource utilization and system elasticity. Aiming at the complex problem of unified management of resource operation in multi-cloud heterogeneous environment, a multi-cloud heterogeneous resource unified management platform is built to realize the global view and unified scheduling of resources. Subsequently, a combined prediction model based on ARIMA-LSTM is proposed, which effectively solves the problem of load prediction accuracy that a single model cannot more accurately perform unified management of multi-cloud heterogeneous resources. At the resource scheduling level, a container scheduling strategy based on Kubernetes is designed, combined with an improved load scheduling algorithm (LSA) to dynamically optimize the container deployment location, and a horizontal elasticity scaling strategy based on deep reinforcement learning is implemented for autonomous decision making in unified management of multi-cloud heterogeneous resources. The results show that this paper's method reduces up to 28.32% resource wastage and 50.75% normalized cost compared to existing resource scheduling algorithms, resulting in an average CPU utilization of 92.8389%, while there is no significant increase in the model's response time. The results prove the effectiveness of the model in this paper in the unified management of multi-cloud heterogeneous resources, providing theoretical support and practical reference for the intelligent use of resources in complex environments.

Index Terms ARIMA-LSTM, Kubernetes, Load Scheduling Algorithm, MDP, Horizontal Elastic Scaling Policy

I. Introduction

With the latest national requirements for the construction of new infrastructure and the rapid development of cloud computing, the demand for cloud-based construction continues to be strong, private clouds, especially government clouds at all levels of government are developing rapidly, and the interconnection and unified management of cloud platforms in a certain region is increasingly reflecting the important status and role [1]-[3]. However, at present, due to the development of various cloud platforms by different manufacturers, although there are relevant cloud computing general standards for the overall specification of the technical framework, the resource management and secondary development interfaces provided by various manufacturers are still somewhat different, resulting in the development of the cloud platform itself is turning into a “chimney type” [4], [5]. Therefore, unified multi-cloud heterogeneous resource management technology has become an emerging trend and urgent need for cloud platform operation and management.

Currently, Hypervisors from various manufacturers are not compatible with each other when it comes to cloud platform operating systems [6]. There is no efficient and unified model for managing multi-cloud heterogeneous resource management [7]. Khatib and Correia [8] proposed a new analytical model for virtual resource management in fully heterogeneous networks, whose key techniques are the estimation of network capacity, the allocation of resources to different Virtual Network Operators (VNOs) and their services, but the allocation and usage of virtual resources are not further explored. Chen, X et al [9] used an innovative architecture to manage various cloud resources with customized models to meet specific management requirements where the operations of the customized models are automatically mapped to the underlying cloud resources through model transformations. Gai, K et al [10] constructed a unified management model for heterogeneous virtual resources in a cyber-physical system with predictive cloud capacity and sustainability factors. Cao, H et al. [11] proposed a service-based framework (SerOri) to address the limitations of the existing one-size-fits-all virtual network resource allocation algorithms for virtual resource allocation in heterogeneous networks. Most of the above research results are

targeted at networks in a specific environment, which makes it difficult to manage heterogeneous resources in a unified way and lacks scalability and generality.

Computational modeling is an important way to improve the resource utilization of cloud infrastructure, and a lot of related work has been done on the management of cloud resource invocation and allocation, and the research mainly focuses on resource management, load balancing, and QoS [12]-[14]. However, many studies have not fully considered the resource overhead, especially the resource utilization in multi-cloud heterogeneous environments should be considered [15]. Holland, O et al [16] proposed a management architecture for aggregating heterogeneous access and spectrum resources of mobile networks to meet the requirements of mobile network capacity and quality of service, but the paper mainly starts from quality of service, and less consideration is given to resource management allocation. Khan et al [17] proposed a hybrid resource orchestrator called HS^2$ to manage heterogeneous cloud resources more efficiently and save energy while minimizing performance impact. Zhao, J et al [18] constructed a heterogeneous computing resource energy consumption minimization and resource management approach based on federated learning algorithms, but the study focuses more on the application in UAV-assisted integrated queueing vehicle networks, which lacks certain applicability. Si, P et al [19] proposed a new cross-network wireless and cloud resource management scheme for heterogeneous mobile cloud network resources that is QoS-aware and maximizes resource tenant revenues while satisfying QoS requirements. Liaskos, C et al [20] proposed a theoretical framework for virtualized programmable meta-surfaces (pm) to enable effective management of cloud heterogeneous resources as 6G communications, thus allowing dynamic deployment of end-to-end services and isolating performance issues. All of these studies proposed effective management schemes for multi-cloud heterogeneous resources, however, the systems are not compatible with each other, making it still difficult to manage all resources in a unified manner [21].

Facing the complex environment of multi-cloud heterogeneous platform, a multi-cloud management platform is constructed to strengthen the unified management of various different cloud resources by launching unified workflow and resource management for multi-clouds. By analyzing the data characteristics of multi-cloud heterogeneous resources, two prediction models, LSTM recurrent neural network and ARIMA, are assembled to establish a prediction model for unified management of multi-cloud heterogeneous resources, and the prediction results of the two models are fused by CRITIC method. After that, this paper models the horizontal elasticity scaling problem as a Markov decision-making process, designs the corresponding state space, action space and reward function by applying multiple indicators related to service quality and resource consumption, and proposes a horizontal elasticity scaling algorithm based on deep reinforcement learning to realize the business requirements while maintaining the controllability and security of the multi-cloud platform.

II. Computational modeling based on unified management of multi-cloud heterogeneous resources

II. A.Multi-Cloud Heterogeneous Resource Unified Management Platform Construction

Aiming at the complex problem of unified management of resource operation in multi-cloud heterogeneous environment [22], we build a multi-cloud management platform by studying the unified service catalog, unified workflow, unified resource management, unified automatic operation and maintenance, unified monitoring and alarm, and API interface of multi-cloud heterogeneous platform.

(1) Unified Service Catalog

Unified service catalog is a key element in the multi-cloud management platform, which can push the functions of the cloud platform to online applications, and can meet the service needs of the cloud platform by customizing the service model, lifecycle policy and many other ways.

(2) Unified Workflow

The multi-cloud management platform has a variety of major roles and workflows that can be individually customized in conjunction with the needs of the application. The multiple roles are mainly maintenance administrators, cloud tenant administrators, system administrators, operation administrators and ordinary users.

(3) Unified resource management

Collect and uniformly store data of multi-cloud heterogeneous platforms, form a unified resource pool, coordinate management, comprehensively determine resource provenance and reasonably allocate resources according to business characteristics and resource allocation rate and utilization rate, and orderly intensify the available resources of multi-cloud management platforms, and at the same time ensure the maximum utilization of multi-cloud heterogeneous resources by way of downgrading and recycling of resources that have been utilized and collected and underutilized in the allocated resources. Multiple measures are taken to realize the unified management of multi-cloud heterogeneous platform resources.

(4) Unified automatic operation and maintenance

The huge number of heterogeneous resources in MultiMultiCloud leads to a huge workload for management personnel, while the expansion of business will further affect the number of heterogeneous resources in MultiMultiCloud. Through unified management, operation and maintenance of cloud platform resources, and automation of cyclical, regular and repetitive work, the system replaces manual labor, which can effectively improve work efficiency and quality.

(5) Unified Monitoring and Alerting

In a multi-cloud environment, the types of multi-cloud heterogeneous resources are complex, which requires the establishment of a multi-cloud monitoring system in order to monitor the actual operation of each cloud platform resource in real time.

(6) Unified API

In addition to providing functions such as resource management and operation services, the multi-cloud management platform should externally provide application program interfaces (APIs) with openness to provide support for third-party cloud platforms to expand their functions on the multi-cloud management platform, so as to be compatible with other cloud platforms and realize interconnection and interoperability of various cloud platforms. The multi-cloud management platform provides APIs that allow it to develop and deploy its own management strategies, thereby rapidly accomplishing the expansion of business functions.

II. B.Multi-cloud load prediction model construction based on LSTM-ARIMA

II. B. 1) LSTM recurrent neural networks

Long Short Term Memory Network (LSTM), is a variant of Recurrent Neural Network (RNN). RNN has shown remarkable results in the prediction of data sequences, however, when the length of the sequence is too large, the RNN leads to a substantial increase in the time of training and may lead to the problem of gradient explosion of the training parameters. Based on the above problems, the LSTM model is proposed, which improves the hidden layer of the RNN, extends the memory function of the network, and enables the model to obtain more persistent information and slows down the decay rate of the information. The core of the LSTM is a memory unit, which consists of forgetting gates, input gates, and output gates, and the structure of “gates” is dependent on the sigmoid activation function, which is the most important function of the network. Depending on the Sigmoid activation function, when the output is 0, the information is discarded, when the output is 1, the information is completely retained, and in other cases the information is partially retained.

i_t is the input gate, f_t is the forgetting gate, o_t is the output gate, c_t is the state of the memory cell at the t moment, x_t is the input vector of the input layer, and h_t is the output vector of the hidden layer. Where the input gate, forgetting gate, output gate, memory cell update state c_t and hidden layer calculation formula are shown in the following equation:

$$i_1 = \sigma(W_{xi}\, x_1 + W_{hi}\, h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_{xf} x_t + W_{hf} h_{t-1} + b_f) \tag{2}$$

$$o_t = \sigma(W_{xo} x_1 + W_{ho} h_{t-1} + b_0) \tag{3}$$

$$c_t = f_t^* c_{t-l} + i_t^* \tanh h(W_{xc} x_t + W_{hc} h_{t-l} + b_c) \tag{4}$$

$$h_t = o_t * \tanh(c_t) \tag{5}$$

where * denotes the element-by-element dot product of the matrix, W_* denotes the weight matrix, b_* denotes the bias matrix, and σ denotes the Sigmoid activation function.

II. B. 2) ARIMA model

Autoregressive Integral Sliding Average Model (ARIMA), is a typical statistical model, the core idea is to consider the predicted time series as a random sequence, to describe that time series with a certain mathematical model approximation, and to predict the value at a certain moment in time based on the determined model. ARIMA model is a combined model, composed as follows equation:

$$(ARIMAp, d, q) = AR(p) + Difference(d) + MA(q) \tag{6}$$

In the formula, AR is the autoregressive model, p is the autoregressive term; Difference is the difference model, d is the difference order; MA is the moving average model and q is the number of moving average terms. The

three parameters, p,d and q, are the key to the order of the model, and the prediction equation of the ARIMA model is as follows:

$$x_t = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \tag{7}$$

where x_t is the sample value, ε_t is the random error disturbance, ϕ_i and ϕ_j are the model parameters, and p,d and q are the model orders. x_t is the multivariate linear function of the first p orders $x_{t-1}, x_{t-2} \cdots, x_{tp}$ and the first q orders $\varepsilon_{t1}, \varepsilon_{t2}, \cdots, \varepsilon_{tq}$.

II. B. 3) Combined LSTM and ARIMA models

A single model prediction cannot be more accurate for the prediction of unified management of multi-cloud heterogeneous resources, so a combined prediction model based on LSTM and ARIMA is proposed [23]. The combination model is to use the LSTM model and ARIMA model to predict respectively after the two predictions are given weights by CRITIC method, the prediction results are weighted cumulative results of the two model predictions, the prediction results are defined in the following equation:

$$Value_{final} = Weight_{LSIM} * Value_{LSIM} + Weight_{ARMA} * Value_{ARIMA} \tag{8}$$

where LSTM model weights and ARIMA model weights are calculated as shown in the following equation:

$$Weight_{LSIM} = \frac{C_{LSIM}}{C_{LSTM} + C_{ARIMA}} \tag{9}$$

$$Weight_{ARIMA} = \frac{C_{ARIMA}}{C_{LSTM} + C_{ARIMA}} \tag{10}$$

$$C_{LSIM} = LSTM_{diff}(1 - r_{LSIM}) \tag{11}$$

$$C_{ARMA} = ARIMA_{diff}(1 - r_{ARIMA}) \tag{12}$$

$$LSTM_{diff} = \sigma_{Sample} - \sigma_{LSTM} \tag{13}$$

$$ARIMA_{din} = \sigma_{Sample} - \sigma_{ARIMA} \tag{14}$$

where C_{LSTM} and C_{ARIMA} denote the information content of the LSTM and ARIMA models, respectively; r_{LSTM} and r_{ARIMA} denote the correlation coefficients of the LSTM and ARIMA models, respectively; and $\sigma_{Sample}, \sigma_{ARIMA}$ and denote the standard deviation of the samples, the predicted results of the LSTM model, and the predicted results of the ARIMA model, respectively.

II. C.Kubernetes-based container scheduling strategy

II. C. 1) Kubernetes Core Concepts

Kubernetes [24] is an open source container orchestration management system that can be used to automate the deployment, management, and scaling of containerized applications with complete cluster management capabilities.Some of the core concepts of Kubernetes are as follows:

1) Pod

In Kubernetes, Pod is the smallest deployable unit of computing. Each Pod contains one or more related containers that share the Pod's network and storage resources and work together to accomplish application-specific functions.

2) ReplicaSet

A ReplicaSet is a set of replicas designed to maintain a set of replicas of a Pod in the Running state, i.e. it ensures the availability of a given number of identical Pods within the platform.

3) Deployment

Deployment is a resource object used to manage the state of an application. It provides declarative update capabilities for Pods and ReplicaSet.

II. C. 2) Kubernetes Architecture

Kubernetes adopts a master-slave distributed structure, there are two roles in the cluster, Master and Node, Master is the control plane of the cluster, responsible for resource scheduling, management and monitoring of the cluster,

the user can interact with the Master node to control the entire cluster through kubectl. Node nodes are the work nodes of the cluster, responsible for completing the tasks assigned by the Master node, and running and managing local containers.The Master node contains four core components: API Server, ETCD, Scheduler and Controller Manager.

II. C. 3) Kubernetes Elastic Scaling

The role of elastic scaling is to improve cloud application resource allocation and ensure cloud application service quality when the load reaches the peak according to the dynamic change of cloud application load. It automatically scales down the cloud application resource allocation during the low peak load period to improve the multi-cloud heterogeneous resource utilization. Elastic scaling can be divided into vertical elastic scaling and horizontal elastic scaling in terms of scaling direction.

II. D.Horizontal elasticity scaling strategy based on deep reinforcement learning

II. D. 1) Research on Horizontal Elasticity Scaling Strategies

In this paper, the improved load scheduling algorithm (SLA) is defined as the upper limit of the application response time, the response time of the application exceeding this upper limit is a violation of the SLA, and the SLA violation rate is calculated as follows:

$$SLA_{vio} = \frac{T_{vio}}{T} \tag{15}$$

where T is the total application running time, and T_{vio} is the time when the response time of the application exceeds the upper limit value defined in the SLA.

For container multi-cloud heterogeneous resources, this paper mainly considers CPU resources and uses the average CPU utilization of application instances to measure resource utilization, which is defined as follows:

$$Avg_{cpu} = \frac{1}{T}\frac{1}{N}\sum_{t=0}^{T}\sum_{i=1}^{N}U_{ti} \tag{16}$$

where N is the number of instances of the application and U_{ti} denotes the CPU utilization of the i th instance at moment t.

II. D. 2) MDP model

A Markov Decision Process (MDP) is a mathematical model describing a sequential decision process, defined by the quintuple (S, A, R, P, γ), where S denotes the state space, the non-empty set consisting of all possible states in the MDP. A denotes the action space, the set of all available actions in the MDP. P denotes the state transfer probability matrix, which describes the probability of the environment transferring to the next state s' after the execution of an action a by an intelligent body in state s. R is the reward function, i.e., the instantaneous reward received by the intelligent body at the time of state transfer. $\gamma \in [0,1]$ is the reward discount factor, the closer to 1 the more the intelligent body focuses on long-term gains. The goal of the intelligent body is to maximize the cumulative reward G_t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \tag{17}$$

In this paper, we model the horizontal elasticity scaling problem as an MDP model where the optimization objective of the intelligences is to maximize the average CPU utilization of the application and minimize the SLA violation rate.The specific definitions of the three key elements of the MDP are as follows:

1) State Space

In order to portray the current application state more accurately and comprehensively, the state space should contain key information related to container level and service level. At time step t, the current state includes the average CPU utilization of the current application Pod U, the number of allocated Pod replicas N, and the current user request rate W. i.e., the state can be represented as:

$$s_t = \{U_t, N_t, W_t\} \in S \tag{18}$$

2) Action space

The action space consists of horizontally scaled operations on applications. At time step t, given a state s_t, the action a_t of an intelligent body can be represented as:

$$a_t = \{-2 \mid -1 \mid 0 \mid +1 \mid +2\} \in A \tag{19}$$

Where ±1 and ±2 denote expanding/shrinking the application by 1 Pod, expanding/shrinking by 2 Pods, respectively, and 0 denotes that no scaling operation will be performed in this decision cycle to avoid additional system overhead due to frequent horizontal scaling operations.

3) Reward function

The design of the reward function should be associated with the optimization objective. In order to ensure the quality of service of the application and maximize the resource utilization, the reward function is defined as shown in equation (20). Where, util is the application's current Pod average CPU utilization, reflecting the application's current level of effective use of CPU resources. perf represents the application's performance performance, which is defined as shown in equation (21). That is:

$$R = util + perf \tag{20}$$

$$perf = \begin{cases} \dfrac{1}{1+\dfrac{RT_{obs}}{RT_{sla}}} & if \quad RT_{obs} < RT_{sla} \\ p & if \quad RT_{obs} \geq RT_{sla} \end{cases} \tag{21}$$

where RT_{sla} and RT_{obs} denote the upper limit of the request response time and the actual response time of the application defined in the SLA, respectively, both expressed in the 95th percentile. p is a negative constant that signals the penalty received by the smart when the SLA is violated due to improper scaling operations.

II. D. 3) Horizontal Elastic Scaling Algorithm Based on Deep Reinforcement Learning

Traditional table-based reinforcement learning algorithms generally use tables to store state-action value functions $Q(s,a)$, which is suitable for simple problems with small state and action spaces. Take Q-learning as an example, its value function update equation is as follows:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)] \tag{22}$$

However, when facing tasks with high-dimensional state space or action space continuum, tabular reinforcement learning methods have the problems of occupying too much storage space and searching for already existing state-action values inefficiently. Based on this, this paper combines deep learning and reinforcement learning algorithms (DQN) to approximate the function of $Q(s,a)$ using a convolutional neural network, and the new value function can be expressed as $Q(s,a;\theta)$, where θ is all the parameters of the neural network. In addition to the estimation network, DQN also introduces another objective network $Q(s,a;\theta^-)$ with the same network structure and different time-step parameters, with the reward function to compute the time-differentiated objective, and updates the parameters of the estimation network by gradient descent θ. Therefore, the network parameter updating formula of DQN can be extended from Eq. (22) as:

$$\theta_{t+1} = \theta_t + \alpha(Y_t - Q(s_t,a_t;\theta_t))\nabla_{\theta_t} Q(s_t,a_t;\theta_t) \tag{23}$$

$$Y_t = R_{t+1} + \gamma \max_a Q(s_{t+1},a;\theta_t^-) \tag{24}$$

In addition, since the interaction data of the intelligences with the environment does not satisfy the independent and homogeneous distribution property, DQN also introduces the experience playback to store the history of the interaction data (s_t,a_t,s_{t+1},r_t) to break the correlation between the data. Based on the above ideas, this paper designs a horizontal elasticity scaler (DScaler) based on deep reinforcement learning. Within the Kubernetes platform, the scaler solves the interaction process of horizontal elastic scaling decisions.

III. Application effect analysis based on multi-cloud heterogeneous resource unified management

III. A. Analysis of results of container scheduling strategy based on container load prediction

Utilizing the monitoring module to realize the collection of container load sample data, this experiment collects 30 minutes of load data to construct a data set, the time interval of data collection is 5 seconds, a total of 360 data, training data for the first 90%, test data for the last 10%. Because the container application deployed in the experimental process of this paper is a computationally intensive application that requires the CPU to do a lot of arithmetic processing, and consumes a high amount of CPU resources, only one type of load, CPU, can effectively

reflect the real load situation of the container application. Therefore, in this paper, we only consider CPU as a load type for prediction analysis, which is used as a source of basis for container scheduling.

The container load time series is shown in Fig. 1. It can be seen that the container load data changes dynamically and randomly, and the load value fluctuates frequently in a short period of time, which leads to the conclusion that the load data contains linear component data and nonlinear component data. The CPU utilization of the sample data is higher in the early period than in the later period because the collected data is collected in the environment under the control of reinforcement-based learning container scaling policy, and the load request is shared by the new Pod, so the container CPU utilization will be reduced, and that is why there is such a trend in the sample data.

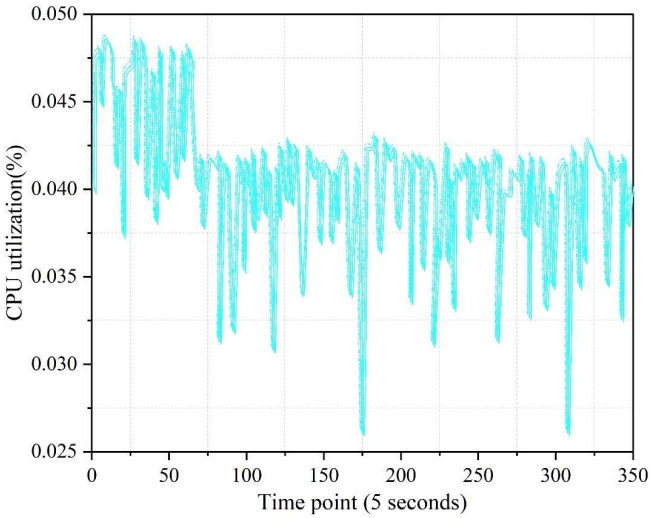

Figure 1: Container load time series

III. A. 1) Load Prediction Experiment

(1) ARIMA model parameterization

Data smoothness testing. Figure 2 shows the ACF autocorrelation plot. The autocorrelation plot leads to the conclusion that the container load time series data is not smooth.

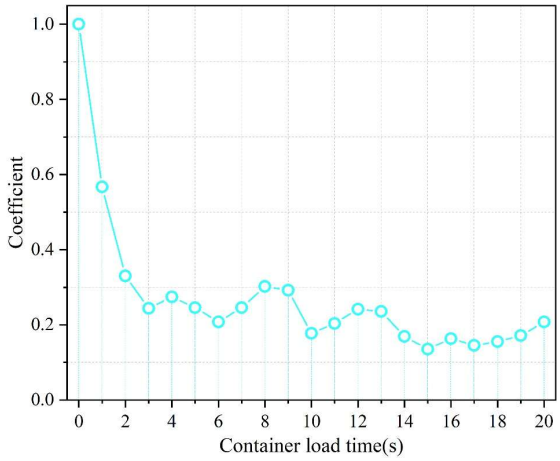

Figure 2: ACF self-correlation diagram

Data Smoothness Processing: the original time series data of container load is done differential processing, and the obtained first-order differential series is shown in Fig. 3.

After the container load time series data is first-order differenced, the first-order differenced data needs to be tested for smoothness, where the autocorrelation plot of the first-order differenced data is shown in Fig. 4, where the presence of a truncated tail indicates that the differenced time series is smooth.

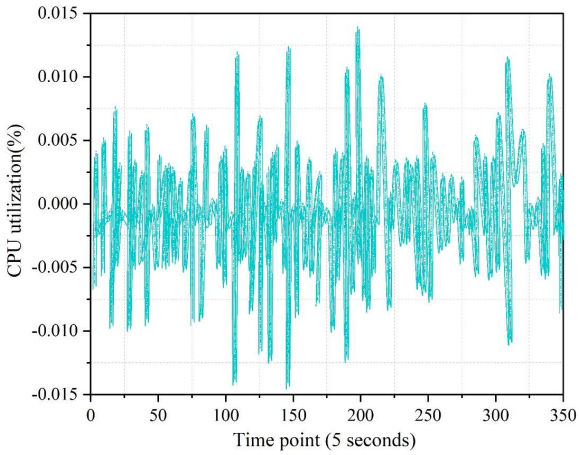

Figure 3: First order difference sequence

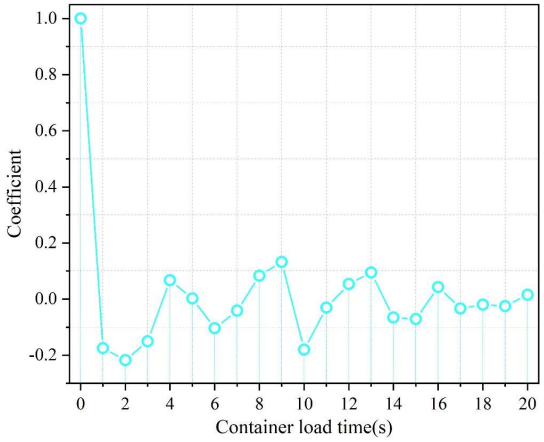

Figure 4: ACF self-correlation diagram

Model ordering: Differencing the non-smoothed data uniquely determines the sequence difference order d. For the p and q parameters, there are many possibilities, here all possible combinations of values between 0 and 4 are used to calculate the information content, and the set of parameters corresponding to the smallest information content is the optimal parameter of the model. Table 1 shows the AIC information content. It can be seen that (1,2) is the smallest and hence three parameters of the ARIMA model are identified as 1,1,2.

Table 1: AIC information

Lags	MA0	MA1	MA2	MA3	MA4
AR0	-2765.9199	-2872.6300	-2894.1359	-2885.3479	-2872.9570
AR1	-2922.3130	-2893.5594	-2929.2007	-2911.0793	-2921.2858
AR2	-2924.4608	-2908.0439	-2914.0618	-2905.3007	-2918.3249
AR3	-2919.4694	-2897.1452	-2914.3392	-2914.6091	-2895.6873
AR4	-2910.9825	-2915.5592	-2898.9733	-2873.8050	-2897.5787

Model test: the parameters of the model were determined through the model fixed order above, the determined parameters were substituted into the model test, by plotting the ACF and PACF of the residual series of the container loading time series as shown in Fig. 5, it can be seen that the model passes the test, and the model was finally determined to be ARIMA(1,1,2).

(2) Load Prediction

The combined prediction model is trained by the experimental data obtained above, and the container load data is predicted using the trained model.The parameter values of the LSTM model parameters are as follows: train_test_split: 0.9, Epochs: 200, and batch_size: 50.The parameter values of the ARIMA model are: p: 1, d: 1, and q: 3.

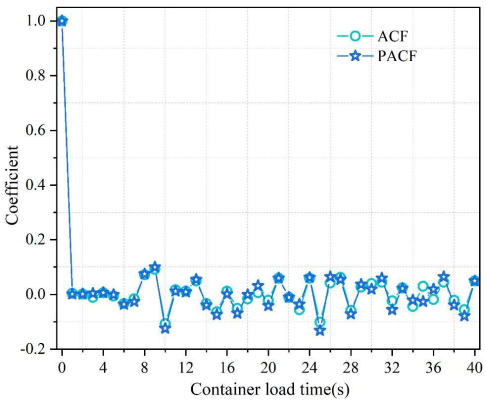

Figure 5: ACF and PACF Diagram

The container load data is predicted using the combined container load prediction model as well as each single model respectively, and the model prediction results are shown in Fig. 6. It can be seen that the predicted values of the combined ARIMA-LSTM prediction model are closest to the true values, and it can be concluded that the combined prediction model predicts the best results.

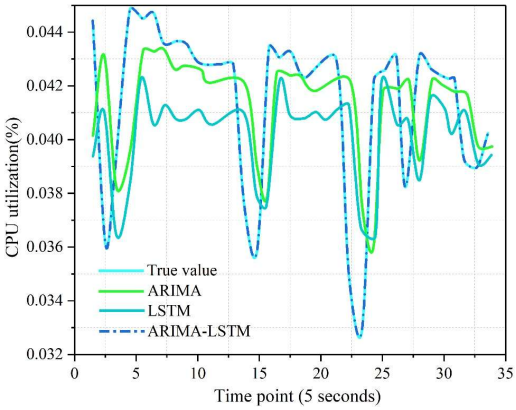

Figure 6: Predictive effect of the model

III. A. 2) Container scheduling experiments

The predicted container load values from above are used as inputs to the container scheduling policy as a selection criterion for the scheduled containers. The experimental results are measured in terms of cluster resource utilization and cluster load balancing degree. The cluster CPU resource utilization is shown in Figure 7.

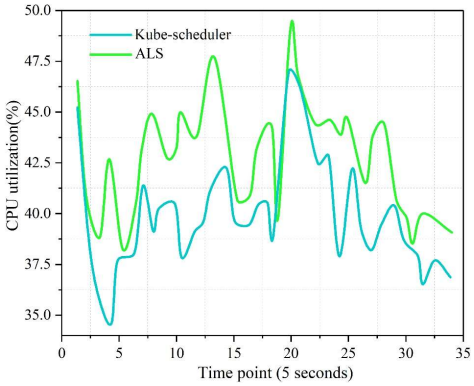

Figure 7: Utilization of cluster CPU resources

Table 2 shows the cluster resource load balancing degree, it can be concluded that the cluster load balancing degree under the SLA policy is lower than that of the Kube-scheduler policy, which indicates that the SLA policy is

more effective in ensuring the cluster load balancing. The SLA policy can improve the cluster resource utilization rate more effectively than the Kube-scheduler. Therefore, the SLA policy is better than the Kubernetes native scheduling policy, and better meets the demand for unified resource management in multi-cloud heterogeneous scenarios.

Table 2: Cluster resource load equalization

Survey content	Kube-scheduler scheduling	SLA scheduling
Load equalization	0.1997	0.0812

III. B. Analysis of auto-scaling effect of unified management of multi-cloud heterogeneous resources

III. B. 1) Comparison Algorithm

To evaluate the performance of ARIMA-LSTM, ARIMA-LSTM is compared with THS, a threshold-based horizontal scaling algorithm, HGRL, a reinforcement learning-based grouping-based horizontal scaling algorithm, and the recent algorithm HVSRL in the literature, which is a reinforcement learning algorithm that performs both horizontal and vertical scaling based on the global state, where the vertical scaling is synchronized across all the instances scaling is synchronized across all instances and is not subdivided into individual instances. In the THS algorithm this paper sets the upper and lower CPU utilization limits to 80% and 40%, respectively, and adds an instance when the global CPU utilization of the application exceeds the upper limit, and removes an instance when the CPU utilization falls below the lower limit. The regular performance of ARIMA-LSTM can be evaluated by comparing it with the most common threshold-based scaling algorithms.HGRL is a degradation algorithm for ARIMA-LSTM that removes the vertical scaling feature, and by comparing it with this algorithm, we can evaluate the enhancement of the vertical scaling for the performance of the application, as well as highlight the role of the vertical scaling in case of localized performance problems.

III. B. 2) Algorithmic effects of different intelligences group capacity (k)

Considering the trade-off between algorithm accuracy and stability, it is necessary to choose an appropriate k. It can be seen that k determines how the logical grouping is performed: if k is large, the algorithm state space and action space are large, the parallelism during learning is small, and the algorithm takes a long time to train, while if k is small, the algorithm contains less information about the application for each training, and the algorithm accuracy will decrease. Moreover, the upper and lower limits n of the horizontal actions are determined by k and the number of instances N. A smaller k may make the system unstable, in extreme cases such as when k=1, each instance forms itself into a group of intelligentsia, it may be that the workload is less in a certain time period, and all the groups learn the leveling action by decreasing one instance, then the application will have no instances in the next time period, and a larger k will make the system too late to make appropriate adjustments in case of drastic changes in the load, and a large number of requests violating the QoS will be generated. Therefore, this paper first compares the effect of the algorithms at different k. The results of the algorithm comparison at different k are shown in Table 3. It can be seen that the algorithm performs better when k=3, 4. The algorithm at k=3 has the lowest QoS violation percentage of 0.4816%, higher CPU utilization of more than 85%, and uses less amount of CPU resources; the algorithm at k=4 has the highest CPU utilization of 86.6458%, and uses the least amount of CPU resources.

Table 3: The algorithm compares the results of the different k

k	Violation of the percentage of QoS(%)	Average CPU utilization(%)	The amount of resources of the average CPU	Average response time(s)
2	2.9995	70.8306	389.3319	0.4548
3	0.4816	85.2715	337.0663	0.4982
4	0.7635	86.6458	323.0384	0.5188
5	1.3567	78.7852	357.2412	0.5647

The results of the comprehensive performance comparison of different algorithms are shown in Table 4.The QoS violation percentage, average CPU utilization, average CPU resources, average response time, and normalized cost of the four algorithms during the experimentsc show that the three reinforcement learning algorithms show significant improvement in QoS violation percentage and average response time compared to THS, which is mainly due to the fact that the experiments start with a The reinforcement learning algorithm handles better when the load increases sharply.ARIMA-LSTM is made to perform optimally in terms of QoS violation percentage, average CPU utilization, and average CPU resources by fine-grained vertical scaling, with 0.4356%, 92.8389%, and 307.0938,

respectively.However, it is not as good in terms of average response time as HGRL and HVSRL, which is due to the fact that HGRL and HVSRL always use more CPU resources and requests can be processed faster under normal circumstances. In terms of normalized cost c, although HVSRL and HGRL each have advantages in the amount of CPU resources and the percentage of QoS violations, the former has a larger c than the latter, mainly because of the extra cost of vertical scaling.ARIMA-LSTM, although it also incorporates vertical scaling, improves significantly in the percentage of QoS violations and the amount of resources used, so ARIMA-LSTM has the smallest c (0.3946).

Table 4: The combination of different algorithms can compare results

k	Violation of the percentage of QoS(%)	Average CPU utilization(%)	The amount of resources of the average CPU	Average response time(s)	c
THS	3.7903	61.9670	428.4020	0.9704	0.8012
HGRL	0.5962	64.9148	422.8615	0.4305	0.6058
HVSR	1.1088	70.2077	401.2866	0.4569	0.7110
ARIMA-LSTM	0.4356	92.8389	307.0938	0.4656	0.3946

IV. Conclusion

This paper proposes a set of load prediction models based on LSTM-ARIMA, which can realize the unified and efficient management of multi-cloud heterogeneous platforms through multiple means in parallel.

The load prediction results show that the combined ARIMA-LSTM prediction model has the best prediction effect, and its prediction results are infinitely close to the real value; the container scheduling strategy based on container load prediction can more effectively ensure the balance of the cluster load, which can effectively improve the utilization rate of the overall resources of the cluster, and better meet the demand for unified scheduling of resources in the multi-cloud heterogeneous scenario.

Multi-cloud heterogeneous resource unified management experiments demonstrate that the amount of applied resources of ARIMA-LSTM combined prediction model is more attuned to the load variations, the system stability is higher, and it performs excellently in guaranteeing QoS and reducing the used resources. The algorithm has the highest CPU utilization and the least amount of CPU resources when the smart body group capacity is 4, with values of 86.6458% and 323.0384, respectively.In addition, ARIMA-LSTM performs optimally in terms of percentage of QoS violation, average CPU utilization, average CPU resources, and normalized cost, with values of 0.4356%, 92.8389%, respectively, 307.0938 and 0.3946.

References

[1] Merseedi, K. J., & Zeebaree, S. R. (2024). The cloud architectures for distributed multi-cloud computing: a review of hybrid and federated cloud environment. The Indonesian Journal of Computer Science, 13(2).

[2] Castañé, G. G., Xiong, H., Dong, D., & Morrison, J. P. (2018). An ontology for heterogeneous resources management interoperability and HPC in the cloud. Future Generation Computer Systems, 88, 373-384.

[3] Pellegrini, A., Di Sanzo, P., & Avresky, D. R. (2016, May). Proactive cloud management for highly heterogeneous multi-cloud infrastructures. In 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 1311-1318). IEEE.

[4] Xu, D., Liu, F., Chen, W., He, F., Tang, X., Zhang, Y., & Wang, B. (2022, May). A review of research on multi-cloud management platforms. In ISCTT 2022; 7th International Conference on Information Science, Computer Technology and Transportation (pp. 1-16). VDE.

[5] Lu, C., Xu, H., Ye, K., Xu, G., Zhang, L., Yang, G., & Xu, C. (2023, May). Understanding and optimizing workloads for unified resource management in large cloud platforms. In Proceedings of the Eighteenth European Conference on Computer Systems (pp. 416-432).

[6] Costache, S., Dib, D., Parlavantzas, N., & Morin, C. (2017). Resource management in cloud platform as a service systems: Analysis and opportunities. Journal of Systems and Software, 132, 98-118.

[7] Danquah, W. M., & Altilar, D. T. (2021). UniDRM: Unified data and resource management for federated vehicular cloud computing. IEEE Access, 9, 157052-157067.

[8] Khatibi, S., & Correia, L. M. (2017). Modelling virtual radio resource management in full heterogeneous networks. EURASIP Journal on Wireless Communications and Networking, 2017(1), 73.

[9] Chen, X., Zhang, Y., Huang, G., Zheng, X., Guo, W., & Rong, C. (2014). Architecture-based integrated management of diverse cloud resources. Journal of Cloud Computing, 3, 1-15.

[10] Gai, K., Qiu, M., Zhao, H., & Sun, X. (2017). Resource management in sustainable cyber-physical systems using heterogeneous cloud computing. IEEE Transactions on Sustainable Computing, 3(2), 60-72.

[11] Cao, H., Xiao, A., Hu, Y., Zhang, P., Wu, S., & Yang, L. (2020). On virtual resource allocation of heterogeneous networks in virtualization environment: A service oriented perspective. IEEE Transactions on Network Science and Engineering, 7(4), 2468-2480.

[12] Guzek, M., Bouvry, P., & Talbi, E. G. (2015). A survey of evolutionary computation for resource management of processing in cloud computing. IEEE Computational Intelligence Magazine, 10(2), 53-67.

[13] Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: a big picture. Journal of King Saud University-Computer and Information Sciences, 32(2), 149-158.

[14] Xu, M., Tian, W., & Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in cloud computing. Concurrency and Computation: Practice and Experience, 29(12), e4123.
[15] Krishnasamy, K. G., Periasamy, S., Periasamy, K., Prasanna Moorthy, V., Thangavel, G., Lamba, R., & Muthusamy, S. (2023). A pair-task heuristic for scheduling tasks in heterogeneous multi-cloud environment. Wireless Personal Communications, 131(2), 773-804.
[16] Holland, O., Aijaz, A., Kaltenberger, F., Foukalas, F., Vivier, G., Buczkowski, M., & Pietrzyk, S. (2016). Management architecture for aggregation of heterogeneous systems and spectrum bands. IEEE Communications Magazine, 54(9), 112-118.
[17] Khan, A. A., Zakarya, M., & Khan, R. (2019). H $^ 2$—A Hybrid Heterogeneity Aware Resource Orchestrator for Cloud Platforms. IEEE Systems Journal, 13(4), 3873-3876.
[18] Zhao, J., Nie, Y., Zhang, H., & Yu, F. R. (2023). A UAV-aided vehicular integrated platooning network for heterogeneous resource management. IEEE Transactions on Green Communications and Networking, 7(1), 512-521.
[19] Si, P., Zhang, Q., Yu, F. R., & Zhang, Y. (2014). QoS-aware dynamic resource management in heterogeneous mobile cloud computing networks. China Communications, 11(5), 144-159.
[20] Liaskos, C., Katsalis, K., Triay, J., & Schmid, S. (2023). Resource management for programmable metasurfaces: Concept, prospects and challenges. IEEE Communications Magazine, 61(11), 208-214.
[21] Marquez, J., Mondragon, O. H., & Gonzalez, J. D. (2021). An Intelligent Approach to Resource Allocation on Heterogeneous Cloud Infrastructures. Applied Sciences, 11(21), 9940.
[22] Krishnasamy Kamalam Gobichettipalayam,Periasamy Suresh,Periasamy Keerthika,Prasanna Moorthy V.,Thangavel Gunasekaran,Lamba Ravita & Muthusamy Suresh. (2023). A Pair-Task Heuristic for Scheduling Tasks in Heterogeneous Multi-cloud Environment. Wireless Personal Communications,131(2),773-804.
[23] Shitian Li,Junzhe Zhang,Ziyu Zhang,Xv Chu,Lili Song & Xiaojun Wang. (2024). Combinatorial Optimisation Model for E-Commerce Retail Merchant Demand Forecasting Based on ARIMA and LSTM. Information Systems and Economics,5(5),
[24] Bogdan Ghimis. (2025). Detecting Interaction Related Bugs in a Multi-Tenant Setting Using Kubernetes. International Journal of Software Engineering and Knowledge Engineering,35(03).