GEORGE BROWN PRESS
JOURNALS, BOOKS & PROCEEDINGS

Journal of Computer, Signal, and System Research

Article

Resource Demand Prediction and Optimization Based on Time Series Analysis in Cloud Computing Platform

Jiaying Huang [1,*]

[1] EC2 Core Platform, Amazon.com Services LLC, Seattle, Washington, 98121, United States
* Correspondence: Jiaying Huang, EC2 Core Platform, Amazon.com Services LLC, Seattle, Washington, 98121, United States

Abstract: In the cloud computing environment, dynamic load changes pose higher requirements for the predictive ability of resource scheduling. To improve resource utilization and reduce the risk of SLA default, this paper proposes a multi-model integrated resource demand prediction framework, combining ARIMA and LSTM to capture linear and nonlinear features in time series data. The framework uses Alibaba Cloud Tianchi load data, including CPU, memory, and network metrics, as learning samples for model training and evaluation; Experiments show that the ARIMA-LSTM hybrid model has better performance in both RMSE and MAE indicators than the single model, with the minimum RMSE being 7.32. To further improve the efficiency of resource allocation driven by prediction, the study introduces the ensemble learning method based on LightGBM, the hierarchical time series analysis mechanism, and the ridge regression dynamic allocation strategy combined with L1 regularization. In the deployment environment of the hybrid cloud platform, this framework has increased resource utilization by 19.6% and reduced service Level Agreement (SLA) default events by more than 50%, mainly due to improved prediction accuracy that minimizes excessive reservations and prediction deviations. The research results provide practical and effective methods and examples for the accurate and secure resource scheduling of cloud service platforms and the implementation of cloud services.

Keywords: cloud computing; resource prediction; time series analysis

1. Introduction

With the increasingly complex business of cloud computing platforms, the uncertainty of required resources and the emergence of multiple peaks at the same time have put great pressure on traditional static based resource allocation methods, which cannot effectively predict future situations and lead to wastage of resource allocation, reducing service reliability and overall system efficiency. Time series analysis is a technical means of continuous dynamic analysis and processing, which can effectively fit and identify factors such as trend, periodicity, and abnormal interference, and is widely used in load forecasting. In response to the complex characteristics of resource behavior on cloud platforms, it is necessary to construct a prediction framework composed of various time series methods to enhance the model's responsiveness to various characteristics and provide relatively stable inputs for resource scheduling algorithms.

2. Overview of Time Series Analysis Methods

Time series analysis technology can simulate sequences with time-series related data by constructing models, analyzing their trends, periods, and anomalous disturbance attributes, and accurately predicting future development trends. The ARIMA model is derived by combining autoregression and moving average structures on the basis of differential stationarity, and is very suitable for analyzing linear and periodic trends, with good

Received: 25 May 2025
Revised: 07 June 2025
Accepted: 23 June 2025
Published: 25 June 2025

interpretability. LSTM is a type of RNN family that can store long-term relevant information by using gating mechanisms, making it more suitable for handling sudden load increases and nonlinear disturbance problems [1]. Given that the activity characteristics of resources on cloud service platforms have both linear and nonlinear features, this paper uses a combination of ARIMA-LSTM for modeling. Firstly, ARIMA is used to describe the main trends, and then the residual sequence is input into LSTM to process nonlinear disturbances, thereby achieving hierarchical modeling and improving accuracy.

3. Resource Demand Prediction and Analysis in Cloud Computing Platforms

3.1. Resource Management Structure of the Cloud Computing Platform

The resource management framework of cloud computing platforms mainly consists of four parts: resource monitoring, predictor, scheduler, and elastic adjuster. The basic part is to use virtualization technologies such as KVM or Docker to transform physical devices into units that can be scheduled (such as vMemory). The platform obtains real-time resource utilization and converts the data into time series data for the predictor to process. The scheduler (such as Kubernetes) calculates the location where the container should be deployed based on predicted data and scheduling algorithms (such as binpack). The elastic regulator will automatically adjust the number of replicas or nodes based on the predicted load. The system requires low latency API connections and high consistency state sharing between the predictor and scheduling system to support resource optimization configuration and response under high dynamic loads.

3.2. Key Factors in Resource Demand Forecasting

Accurate prediction of resource demand depends on whether the main influencing factors in modeling are fully captured. For example, daily working hours, holidays, etc. can have a huge impact on resource scheduling; Different task types also have different resource usage patterns. For example, the CPU and memory usage of web services is much higher than that of big data processing; User behavior activities (such as session intervals, dwell time, etc.) also need to be considered [2]. Therefore, when modeling, it is necessary to construct time-lagged features using sliding windows, introduce statistical indicators such as mean, variance, and extreme values, and add external triggering conditions (such as software upgrades) to enhance the sensitivity of the model.

3.3. Application of Time Series Analysis in Resource Demand Forecasting

When predicting cloud platform resources, time series analysis techniques can transform variable resource information into function sequences through modeling, in order to extract patterns and make predictions. The ARIMA algorithm performs differential operations on the data to make the sequence stationary, and then uses autoregression and moving average to establish a linear model. However, the LSTM network based on RNN introduces a gating mechanism, which can retain long-term dependencies to capture nonlinear changes. In practice, the platform collects sample data from the CPU or memory according to a fixed time window (such as every half hour) as input for the prediction model, forming a feature matrix. At the application stage, the model is encapsulated in microservice form and continuously receives new data through APIs to predict future short-term resource requirements (such as 5-minute CPU usage), reducing SLA default risk.

3.4. Characteristics and Patterns of Cloud Computing Resource Requirements

The demand for cloud computing resources presents characteristics such as instability, high frequency, and diversity, which require models to technically characterize them. From a resource perspective, parameters such as CPU, RAM, hard disk I/O, and network bandwidth often exhibit different temporal characteristics and require separate modeling; From a temporal perspective, the periodicity of days/weeks is often disrupted by irregular

events, such as a large influx of users, which increases the difficulty of prediction; Some resources have intermittent peak distributions, which make traditional mean methods unable to accurately characterize them; Due to user behavior driven factors, there is often a superposition of multi-source changes, such as the simultaneous increase of I/O and CPU, which requires multiple time series to jointly model.

4. Resource Demand Forecasting Model Based on Time Series Analysis

4.1. Selection and Application of Time Series Analysis Methods

When building a resource demand forecasting model, select appropriate modeling methods based on the characteristics of cloud platform load data. Usually, resource sequences contain long-and short-term trends and cyclical components, making them more suitable for linear fitting using ARIMA models; Due to the influence of user behavior, resource sequences exhibit nonlinear characteristics, making them more suitable for modeling using LSTM models [3]. This article uses the ADF testing method to verify whether the sequence satisfies first-order differencing stationarity. It uses ARIMA (p, 1, q) modeling for trend mining, and adjusts parameters using AIC and residual white noise tests.

In practical applications, ARIMA is used to establish a linear model for resource sequences, and its residuals are used as LSTM inputs to process nonlinear disturbance terms, forming a hierarchical prediction model that improves accuracy and model reliability. The cascading model is deployed as microservices in containers, which can simultaneously perform predictive calculations for multiple resources and achieve automated data flow interaction through APIs. The predicted results serve as inputs for the scheduler, optimizing dynamic resources and improving response speed and resource utilization efficiency.

4.2. Acquisition and Processing of Datasets

In order to build a practical resource demand prediction model, this article uses a dataset from key resource indicators such as CPU utilization, memory usage, and network throughput provided by Alibaba Cloud Tianchi. The data sample has a duration of 30 days and a time granularity of 1 minute. The data is sampled and recorded according to resource nodes, with each record containing a time point, resource sequence number, and resource related values. This study uses scripts to automatically collect data and store it in CSV format. In the preprocessing stage, unnecessary information and nodes will be filtered out or removed to ensure the remaining data are complete and valid. At the same time, the extreme value data will be identified and processed using a 3-fold standard deviation method, and the missing parts will be filled forward according to the chronological relationship to avoid affecting temporal continuity.

Using the linearization normalization method, all feature values are extended to the range of [0,1], expressed mathematically as:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

Among them, x represents the initial feature value, and x_{min} and x_{max} respectively represent the minimum and maximum values in the corresponding training set. This method can ensure stable feature distribution, improve the training convergence rate and prediction stability of the model in different feature dimensions.

4.3. Training and Validation of the Predictive Model

The ARIMA-LSTM cascade model is based on a staged training mode, which includes sample construction, model building, defining loss functions, and designing validation systems. Use ARIMA to model the linear and periodic features in the original sequence, with the residual sequence as the input for LSTM. ARIMA uses ADF testing and AIC standards to determine parameters, ensuring that the residual sequence is stable and learnable [4]. LSTM consists of two LSTM layers, each with 128 hidden units, and uses tanh as the activation function. Choose Adam as the optimizer, with an initial learning

rate of 0.001, a batch size of 64, and 100 training iterations. The loss function adopts mean square error:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \quad (2)$$

Among them, $\hat{y}_i, y_i$ represents the predicted value and the true value, and N represents the number of samples. To prevent overfitting of the model; Dropout is introduced during the model training process; Use EarlyStop to stop iterating when the validation loss has not decreased further. We split the validation data in chronological order to prevent future information leakage. This model is run on the TensorFlow platform for use in prediction scheduling systems.

4.4. Accuracy Evaluation of the Model

The accuracy evaluation of models is the main content of resource prediction systems, which is related to the reliability of scheduling strategies and the completion of scheduling. The roots mean square error, mean absolute error, and mean absolute percentage error are used to evaluate the error magnitude, stability, and relative accuracy of the ARIMA-LSTM model in predicting resource demand.

If there are N samples in the test set, the model output is $\hat{y}_i$, and the actual observation value is y_i, then the three indicators are defined as follows:

$$\text{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2};$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|;$$

$$\text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i + \epsilon}\right| \quad (3)$$

In the formula, ϵ is a small positive number used to prevent the denominator from being zero. The three measures assess the performance of the model in terms of numerical accuracy, volatility sensitivity, and business interpretability. In addition, for multi-dimensional schedulers with multiple resource dimension attributes such as CPU, memory, and network, an independent evaluation strategy is applied individually, with weighted average error used as the overall evaluation metric.

5. Optimization Strategies for Resource Management in Cloud Computing Platforms

5.1. Adopt the Ensemble Learning Method

In order to improve the stability and universality of resource demand prediction, this paper introduces the integrated learning method based on ARIMA-LSTM to overcome the limitations of a single model. In the integrated framework, LSTM is used as the basic prediction model, and LightGBM is introduced as the secondary learner to construct a stacking ensemble model, which can combine the outputs from different base models.

LightGBM has the characteristics of optimizing gradient direction and leaf-wise splitting, so it performs well in handling high-dimensional sparse input features and is superior in dealing with behavioral changes of complex resources. During the training phase, the LSTM model first outputs preliminary predicted values $\hat{y}_t^{(1)}, \hat{y}_t^{(2)}, \ldots, \hat{y}_t^{(K)}$, and then construct a fusion function using LightGBM:

$$\hat{y}_t^{\text{final}} = \sum_{k=1}^{K} w_k \cdot \hat{y}_t^{(k)} + b \quad (4)$$

Among them, w_k is the weight coefficient of each base model, and b is the bias term. LightGBM can independently determine the allocation of weights in the model, allowing it to perform better in unstable and volatile parts.

5.2. Construct Multi-Level Time Series Models

Based on changes in resource activity patterns at different time scales, the multi-level time series analysis method can summarize various change patterns. Based on this, a high-

precision sequence structure can be constructed to enhance the model's ability to identify multi-frequency information.

The model is designed with multi-channel inputs, using time windows (15 minutes, 1 hour, 4 hours) as inputs for different LSTM "channels", extracting time features at different levels, and integrating them into a fully connected layer for output. Assuming the predicted value of the *j*-th time window is $\hat{y}_t^{(j)}$, the total output can be defined as:

$$\hat{y}_t = \sum_{j=1}^{n} \alpha_j \cdot \hat{y}_t^{(j)} \tag{5}$$

Among them, α_j is the weighting coefficient for each time level, which is automatically optimized through backpropagation.

A multi-level structure can enable the model to better adapt to load changes at different granularities, adapt well to periodic disturbances such as large fluctuations during holidays and day night alternation, and effectively improve the scheduling system's fault tolerance for prediction errors.

5.3. Use Multivariate Time Series Analysis Methods

There is a high correlation among multiple resource dimensions such as CPU, memory, disk IO, and network traffic, which exhibit interactive and synchronous fluctuations [5]. In order to better reflect the correlation between various resources, this paper models based on multivariate time series, uniformly representing the dimensions of various resources as multidimensional input tensors, and embedding multivariate LSTM units in the model to achieve parallel modeling.

Let the multidimensional resource state vector at time *t* be:

$$\mathbf{x}_t = [x_t^{\mathrm{CPU}}, x_t^{\mathrm{Mem}}, x_t^{\mathrm{IO}}, x_t^{\mathrm{Net}}] \tag{6}$$

Take the corresponding inputs of *L* consecutive time steps as a three-dimensional tensor $X \in \mathbb{R}^{B \times L \times F}$, define *B* as batch size, *L* as time window length, and *F* as feature dimension. The LSTM network receives input tensors and updates all hidden state information, simulating dynamic correlation processes covering different resources, overcoming the problem of insufficient response to systematic fluctuations in single variable prediction, and enhancing the flexibility of prediction driven scheduling.

5.4. Apply Regularization Techniques

In multidimensional resource prediction, input data often has redundancy or low correlation. Without effective structural control measures, it may lead to problems such as overfitting due to redundant model parameters and decreased generalization ability. In this regard, this article introduces the L1 regularization technique, which utilizes sparsity to constrain network parameters to improve the simplicity and noise resistance of the model structure. That is to say, the L1 regularization term is added to the loss function of the fully connected prediction layer, which is defined as follows:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{d} |w_j| \tag{7}$$

Among them, λ is the parameter that controls regularization, and w_j represents the jth weight of the model. L1 regularization has the ability of automatic feature selection, which can select important features and suppress weak signals, providing a solid input foundation for subsequent scheduling and control.

6. Empirical Research and Case Analysis

An experimental system was designed based on a simulated hybrid cloud platform, equipped with an 8-core CPU, 32GB of memory, and CentOS 7.6 operating system. Docker resource virtualization and Kubernetes management were implemented on this system. The deployment of this experimental environment was achieved by utilizing monitoring log data resources from real machines on the Alibaba Cloud Tianchi Competition official

website. Implement ARIMA-LSTM model using TensorFlow tool, encapsulate the model with microservice software architecture, and connect to the scheduler using RESTful API.

The predictive performance of each model was evaluated based on RMSE, MAE, and MAPE, and the results are compared in Figure 1:

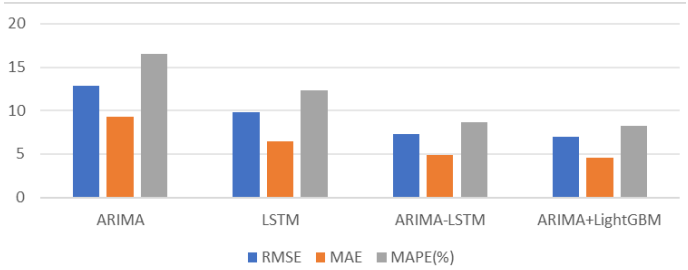

Figure 1. Comparison of Prediction Performance of Different Models.

It can be seen from the data in the figure that when ARIMA and LSTM methods are used comprehensively, the performance of each part is better than that of the single model structure, especially the MAPE index decreases significantly, indicating that this method has better ability to control relative errors. After incorporating LightGBM ensemble learning, the overall error is further reduced.

Based on the model prediction results, the scheduler dynamically adjusts the number of Pod replicas. The performance of static strategy and predictive driven strategy in terms of CPU utilization and SLA default frequency is shown in Table 1 below:

Table 1. Comparison of Scheduling Effectiveness between Predictive Driven and Static Strategies.

Scheduling Strategy	Average CPU Utilization Increased	SLA Default Rates Are Down
Static Strategy	Base Line Value	Base Line Value
Predictive Driving Strategy	↑19.6%	↓51.2%

The predictive auxiliary scheduling scheme significantly reduces resource redundancy and service failure events, and has practical significance.

The experimental results have verified the stability and prediction accuracy of the proposed model in the face of large-scale changes in resource demand. Ensemble learning and regularization mechanisms can enhance the generalizability of models. The scheduling system integrates the predicted results through APIs, achieving dynamic adjustment based on data-driven methods while ensuring service quality. This provides a practical foundation for intelligent scheduling of cloud platforms.

7. Conclusion

This study conclusively demonstrates the efficacy of a multi-model framework integrating ARIMA, LSTM, and LightGBM-based ensemble learning for accurate cloud resource demand forecasting. The model achieved a minimum RMSE of 7.32 and significantly outperformed single-model approaches across RMSE, MAE, and MAPE metrics. The incorporation of hierarchical time series analysis and L1 regularization further enhanced model robustness against volatile workloads. Deployment on a hybrid cloud platform validated tangible operational improvements, including a 19.6% increase in resource utilization and over 50% reduction in SLA violation events through proactive, prediction-driven resource optimization. This establishes a scalable and data-driven methodology for balancing computational efficiency with stringent QoS requirements in dynamic cloud environments. Future research will focus on extending this framework to multi-tenant heterogeneous infrastructures and real-time edge-cloud coordination.

References

1. S. Pang, et al., "Joint trajectory and energy consumption optimization based on UAV wireless charging in cloud computing system," *IEEE Trans. Cloud Comput.*, vol. 11, no. 4, pp. 3426–3438, 2023, doi: 10.1109/TCC.2023.3288527.
2. M. Smendowski and P. Nawrocki, "Optimizing multi-time series forecasting for enhanced cloud resource utilization based on machine learning," *Knowl.-Based Syst.*, vol. 304, p. 112489, 2024, doi: 10.1016/j.knosys.2024.112489.
3. N. Sanjay and S. Sreedharan, "Hybrid bio-inspired optimization-based cloud resource demand prediction using improved support vector machine," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 1, 2024, doi: 10.14569/ijacsa.2024.0150177.
4. R. Rathinam, et al., "SJFO: Sail jelly fish optimization enabled VM migration with DRNN-based prediction for load balancing in cloud computing," *Netw. Comput. Neural Syst.*, vol. 35, no. 4, pp. 403–428, 2024, doi: 10.1080/0954898X.2024.2359609.
5. E. Kholdi and S. M. Babamir, "Reserve policy-aware VM positioning based on prediction in multi-cloud environment," *J. Supercomput.*, vol. 80, no. 16, pp. 23736–23766, 2024, doi: 10.1007/s11227-024-06349-6.