ARIMA-Based and Multiapplication Workload Prediction With Wavelet Decomposition and Savitzky–Golay Filter in Clouds

Jing Bi, *Senior Member, IEEE*, Haitao Yuan, *Senior Member, IEEE*, Shuang Li, Kaiyi Zhang, Jia Zhang, *Senior Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Current cloud data centers (CDCs) provide highly scalable, flexible, and cost-effective services to meet the performance needs of emerging applications. It is critical for CDC providers to predict future incoming workloads such that they can perform accurate resource provisioning in CDCs. Prediction accuracy is important and its improvement has been pursued in much existing work. This work adopts two different real-life Google data traces, based on which such prediction is conducted. Specifically, this work first gives a novel prediction mechanism that integrates wavelet decomposition, Savitzky–Golay (SG) filter, and autoregressive integrated moving average (ARIMA) to realize workload prediction in each time interval. The time series of the workload is smoothed with an SG filter and further divided into several components with wavelet decomposition. Then, an integrated approach is developed to predict statistical trends and their detail components. Real-life trace-driven experiments are done and the results suggest that the proposed method provides higher accuracy of prediction than its existing peers.**

***Index Terms*—Autoregressive integrated moving average (ARIMA), cloud computing, data centers, wavelet decomposition, workload prediction.**

I. Introduction

CLOUD data centers (CDCs) aim to provide highly scalable, cost-effective, and flexible services to meet performance requirements for users. A growing number of cloud applications of users have been developed and efficiently executed in CDCs. Users' data are enabled to be kept and analyzed in third-party CDCs [1], [2]. To achieve it, CDC providers have to fast allocate the optimal number of infrastructure resources to do users' tasks. Therefore, workload preprocessing and prediction are highly needed for dynamic resource provisioning [3], [4], [5], which relies on multiple factors, e.g., user count, future events, past resource usage, and system states. In addition, accurate prediction of future workload can further improve the performance related to energy consumption and resource utilization. Then, operation cost of CDCs, and violations of service level agreements (SLAs) can be reduced greatly [6].

Nevertheless, it is challenging to precisely predict future workload due to nonlinear patterns in users' tasks. Currently, many existing studies design different workload prediction approaches [7], [8], [9], [10], [11], [12], [13], [14]. They propose several prediction models for different types of workload for networks, server clusters, and high-performance platforms [7], [8], [9]. The average or maximum number of tasks for a given time interval is measured accordingly. Nevertheless, they mainly focus on the analysis of these tasks executed in data centers in a statistical way and fail to provide high-quality prediction of future tasks. The workload of CDCs changes rapidly, and therefore, it is difficult to accurately capture its characteristics due to large magnitude differences, and existence of noise. The amount of data we collect from CDCs is not suitable for deep learning, which usually requires enormous training time to obtain a high-quality prediction model. It is worth noting that the training time is also a critical consideration for CDC providers in addition to prediction accuracy. Different from the existing studies, this work proposes an integrated prediction approach to better understand the characteristics of tasks and increase the accuracy of prediction. It adopts past tasks as training data to predict the future tasks in CDCs. It analyzes two different real-life tasks collected from a realistic production data center of Google [15].

The data collected by CDCs contain salt and pepper noise, which can be removed without distorting a shape of the original data with a Savitzky–Golay (SG) filter. The workload request arriving rate we have collected from CDCs is not large enough to train deep learning models. Therefore, autoregressive integrated moving average (ARIMA) is adopted because of its easy training and implementation. By considering the complexity of signals contained in data, we adopt a method

Manuscript received 26 July 2022; revised 21 May 2023; accepted 14 December 2023. Date of publication 10 January 2024; date of current version 19 March 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62173013 and Grant 62073005; in part by the Beijing Natural Science Foundation under Grant 4232049; and in part by the Fundamental Research Funds for the Central Universities under Grant YWF-23-03-QB-015. This article was recommended by Associate Editor H. Tianfield. *(Corresponding author: Haitao Yuan.)*

Jing Bi, Shuang Li, and Kaiyi Zhang are with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn; lishuang2018@bjut.edu.cn; kaiyizhang@aliyun.com).

Haitao Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn).

Jia Zhang is with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX 75205 USA (e-mail: jiazhang@smu.edu).

MengChu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2023.3343925.

Digital Object Identifier 10.1109/TSMC.2023.3343925

2168-2216 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

of wavelet decomposition to eliminate the data noise. It can decompose irregular data into multiple sequences with certain rules. Decomposed signals with fixed information from the original signal are more suitable for training prediction models. The proposed approach predicts future tasks in the next time interval. It adopts a Savitzky–Golay filter, wavelet decomposition [16] (SW), and ARIMA [17], thus leading to SWARIMA to predict future tasks in CDCs. This work aims to make the following new contributions.

1) We compare different sequence smoothing approaches to remove noise points in realistic task data and conclude that the SG filter achieves the best result.
2) We adopt the wavelet decomposition to analyze the detail information of data at multiple resolutions, based on which a novel prediction method is constructed. More specifically, we adopt the Symlet wavelet with more orthogonality and symmetry to achieve better results.
3) We design SWARIMA for producing a prediction model for task data, and the best parameters of this model are obtained according to the Akaike information criterion (AIC).

We evaluate SWARIMA on real-life task data, and compare it with several state-of-the-art peers [11], [18], [19]. Experimental results demonstrate that it outperforms them in terms of various evaluation criteria.

Section II briefs related work. Section IV introduces the data preprocessing of the Google cluster workload traces 2011 and 2019. Section V describes the proposed method. Section VI shows the results of experiments. Section VII summarizes this work and points out future work.

II. Related Work

A. Integrated Algorithms-Based Prediction

Many studies focus on the prediction of workload in CDCs. Classical approaches include hidden Markov model (HMM), ARIMA, and auto-regression (AR). The work in [20] designs a multidimensional and layered HMM to manage time-limited streaming applications. Based on predicted results, a framework is proposed to realize efficient resource scaling for big data applications in clouds. The work in [11] designs a model to capture relationships among virtual machines by using predicted resource requirements with ARIMA. Then, it is used to investigate utilization volatility of resources, and an algorithm for virtual machine placement is designed accordingly. Saxena et al. [8] designed a framework of resource management, which predicts the resource utilization of servers and achieves load balance. An online prediction system of resources is developed in virtual machines to minimize SLA violations and performance degradation. Huang et al. [21] used seasonal ARIMA to predict and analyze the traffic of message passing interfaces in virtual clusters. It allows users to investigate patterns of iterative movement of data without knowing programming details of simulations. Then, a cross-layer middleware system is developed to dynamically schedule network resources in virtual clusters. Fanjiang et al. [12] applied genetic programming to evolve multiple predictors, each of which is used to forecast future quality of service (QoS). Two techniques, including hybrid evolution and elite individual composition, are proposed to improve the prediction accuracy of multistep-ahead time series of QoS. Xie et al. [22] proposed a hybrid prediction model by integrating ARIMA and triple exponential smoothing. It predicts both linear and nonlinear relations in the container workload sequence built in a cloud platform. Sun et al. [23] combined a support vector machine (SVM) with a backpropagation neural network (BPNN) by using the optimal weighting rule for predicting workload. It effectively tackles shortcomings of models, including SVM and BPNN, and improves the prediction accuracy of physical machines in clouds.

However, it is difficult and inaccurate to model the original data of CDCs with the aforementioned methods. Table I provides the results of comparing this work with them. Although some studies consider data smoothing, they do not consider data filtering and decomposition to improve the prediction accuracy. Different from these existing studies, we investigate short-time span changes of tasks and achieve timely prediction. The proposed method is very fast and suitable for realistic CDCs that can quickly execute tasks with the optimal number of resources. This work also applies feedback from newly collected tasks to update the proposed model running in CDCs. SWARIMA well avoids the occurrence of one-step delay and achieves better performance than ARIMA.

B. Machine Learning-Based Prediction

Different machine learning methods are used to predict the time series [24]. They mainly include the variants of prediction models, e.g., convolutional neural networks [25], long short-term memory (LSTM) [26], [27], BPNN [18], and deep belief networks [28]. Singh et al. [10] gave an evolutionary quantum neural network for predicting workload in CDCs. It uses the efficiency of quantum computing with encoded workload information and predicts resource or workload demands for higher accuracy. Feng et al. [13] presented an ensembling model to predict workloads by integrating an adaptive sliding window and the time locality. The sliding window mechanism jointly integrates trend and time correlations, and random workload fluctuations for online regression, and yields higher accuracy with lower overhead. A time locality mechanism is proposed for behaviors of local predictors and a multiclass regression approach for integration of models is designed. Ding et al. [14] adopted online learning and transfer learning to design an integrated forecasting model for the prediction of container workloads, thereby providing higher adaptivity, availability and generality. The work in [18] provides a self-adaptive differential evolution method to train a neural network-based workload prediction model. It obtains high prediction accuracy for workload bursts by determining the optimal crossover rate. Zhang et al. [29] proposed a proactive prediction algorithm for Docker container workloads by integrating LSTM and triple exponential smoothing. They investigate both long-term and short-term dependencies in the time series of container resources and remove data noise in container resource utilization. Tuli et al. [19] used LSTM to realize the average workload prediction over successive time

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON AMONG THIS WORK AND ITS STATE-OF-THE-ART STUDIES

Reference	Data smoothing	Small data set	Data filtering	Data decomposing	Fast training
[10]	×	×	×	×	×
[11]	×	✓	×	×	✓
[12]	×	✓	×	×	✓
[13]	×	✓	×	✓	✓
[14]	×	✓	×	×	×
[18]	×	✓	×	×	✓
[19]	×	×	×	×	×
[20]	×	✓	×	×	✓
[21]	×	✓	×	×	×
[22]	×	✓	×	×	✓
[23]	×	✓	×	×	✓
[24]	×	✓	×	×	✓
[30]	×	×	×	×	×
[31]	×	✓	×	×	×
[32]	×	×	×	×	×
This work	✓	✓	✓	✓	✓

intervals in the future, and adopt two real-life data to evaluate its performance. Lu et al. [30] presented a forecasting model by combining a K-means algorithm and BPNN with random learning rates to predict the arrival trend of future workload demands. It exploits unclear characteristics of latency sensitivity of dynamic cloud workloads. Gupta et al. [26] adopted a gradient descent mechanism and a Levenberg–Marquardt adaptation method to predict dynamic resource utilization. An online multivariate sparse framework is proposed to realize the fast prediction of resource usage in a cloud.

Most of the above-mentioned workload prediction approaches use nonlinear regression and neural networks, and are difficult to cope with irregular and catastrophic bursts of workload. Most state-of-the-art studies do not consider the stationarity of data. These methods do not adopt data filtering and decomposition to investigate data features, thus disenabling them to well handle fast-changing workloads. Most studies require much training data, thereby demanding substantial training time. Besides, deep learning approaches demonstrate their good performance in large-scale workload prediction for CDCs. However, their training time is usually long when they are used to deal with large-scale data. To evaluate the performance of SWARIMA, this work adopts two different real-life Google workload traces within almost one month in Google production computing clusters, and it is collected from a single Borg cell with 12 500 machines. The total size of the dataset is approximately 41 GB. However, we find that the data about task arriving rates is suitable for evaluating the performance of classical prediction methods. The scale of data about task arriving rates is not large enough to well train deep learning models, which often need much more data for their training. In addition, we conduct many experiments and find that given the Google workload traces, classical methods perform better than deep learning ones in terms of workload prediction accuracy. However, as discussed above, there are several limitations in many existing classical methods.

An SG filter can remove salt and pepper noise in the original data, which makes features of the sequence easy to analyze. Wavelet decomposition methods can decompose irregular data into several regular sequences. The decomposed data are helpful to the construction of prediction models. Wavelet decomposition can also improve the smoothing of data and reduce its size, which is beneficial to ARIMA's performance. ARIMA is widely used in time-series analysis because it has fewer parameters and is easy to train. Therefore, this work proposes SWARIMA as an integrated machine learning approach to achieve the prediction of workload in CDCs.

III. PROBLEM FORMULATION

This work aims to accurately predict the number of arriving tasks in the next time slot in CDC based on historical data. In time slot $k+1$, tasks in previous k time slots provide the input I ($\{I_1, I_2, \ldots, I_{k-1}, I_k\}$). We need to predict the number of tasks ($\hat{y}_{k+1}$) in time slot $k+1$, i.e.,

$$\hat{y}_{k+1} = f(I_1, I_2, \ldots, I_{k-1}, I_k). \tag{1}$$

The objective of training the proposed model is to minimize the difference between $\hat{y}_{k+1}$ and its ground-truth one (y_{k+1}).

IV. DATA PREPROCESSING

A. Data Normalization

To obtain the clear attributes of arriving workloads, we adopt and investigate two different real-life Google cluster

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

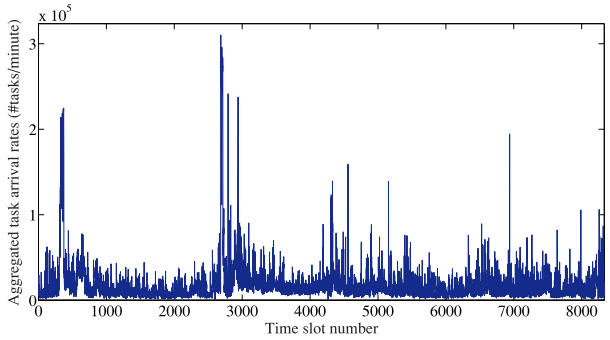


Fig. 1. Aggregated task arriving rate in Google workload trace 2011.

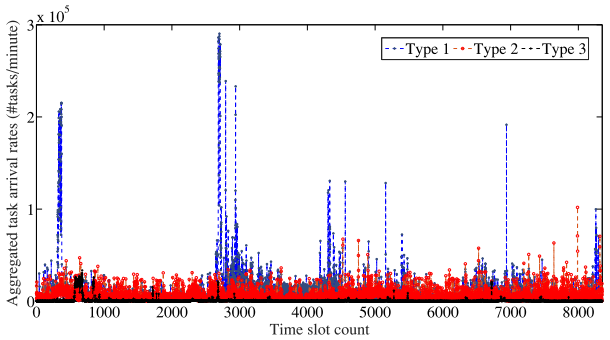


Fig. 2. Aggregated task arriving rate of each type in Google workload trace 2011.

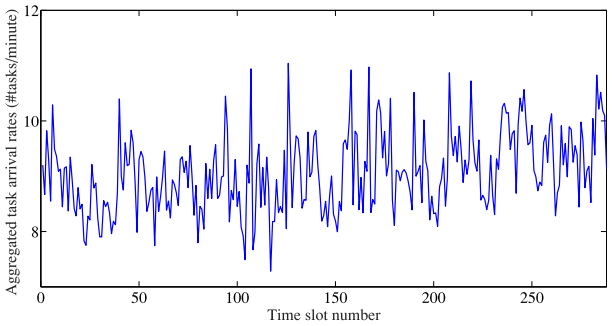


Fig. 3. Aggregated task arriving rate (log) in Google workload trace 2011.

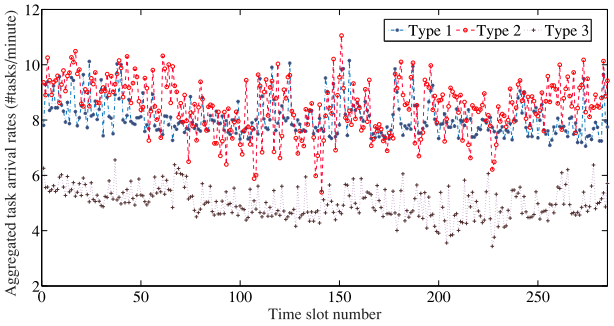


Fig. 4. Aggregated task arriving rates of different types (log) in Google workload trace 2011.

workload traces over a month-long period in 2011[1] and 2019.[2] In the Google workload trace 2011, each task represents an attribute to indicate its importance. We group them into three types: gratis (0–1), other (2–8), and production (9–11) in the 12 priorities [31]. Here, similar to [32], it is assumed that resource requirements of tasks corresponding to each type of applications are similar. For each group of tasks, the period of 29 days is divided into 8352 time intervals. The length of each time interval is five minutes. We calculate the number of aggregated tasks in each time interval illustrated in Fig. 1. Furthermore, Fig. 2 illustrates the aggregated task arriving rate of each type in Google workload trace 2011. Similarly, more details about the Google cluster workload trace 2019 can be found [33].

To decrease the standard deviation of the actual workload, we first perform the logarithmic operation on the task arriving rates. For example, in Figs. 1 and 2, the number of arriving tasks for the Google workload dataset 2011 varies significantly. Consequently, we standardize the workload to effectively eliminate the heteroscedasticity of the actual workload. After the above process, we can obtain the workload with a maximum fluctuation of 3.5208 and the average of 0.2347. As shown in Figs. 3 and 4, after the normalization, the total aggregated task arriving rate and that of each type on the 25th day are relatively smooth.

B. SG Filter

The workload time series needs to be smoothed, and therefore, noise must be removed. To provide better prediction results, this work adopts moving median (MM), moving average (MA), and SG filters to remove noise from the original workload. The results show that the SG filter with a window size of five achieves the best performance among these tested ones.

The SG filter removes the noise from the original workload (X), i.e.,

$$X = \{x_1, x_2, \ldots, x_t\}, \quad t \in N^+ \tag{2}$$

where N^+ denotes a set of positive integers, and x_t denotes the number of tasks arriving in time interval t.

Y denotes a subsequence of X with the size of $2m+1$, i.e.,

$$Y_k = \{x_{k-m}, \ldots, x_k, \ldots, x_{k+m}\}, \quad k \in [m+1, t-m]. \tag{3}$$

In this work, $m = 2$ and $R = 3$. Polynomial $p(n)$ is obtained as

$$p(n) = \sum_{r=0}^{R} a_r n^r, \quad n \in [-m, m]. \tag{4}$$

$\mathcal{E}$ denotes the error of mean-squared approximation of the subsequence, which is obtained as

$$\mathcal{E} = \sum_{n=-m}^{m} (p(n) - x_{k+n})^2 = \sum_{n=-m}^{m} \left(\sum_{r=0}^{R} a_r n^r - x_{k+n} \right)^2. \tag{5}$$

We choose a central data point of the obtained polynomial as the smoothed one in each time interval. Then, the least-square estimation is used to update parameters for minimizing $\mathcal{E}$ [34].

[1] https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md
[2] https://github.com/google/cluster-data/blob/master/ClusterData2019.md

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

TABLE II
ADF TEST RESULTS

Types	Values
ADF	-9.18045751
p value	2.25931e-15
Critical value (10%)	-2.56695506
Critical value (5%)	-2.86188766
Critical value (1%)	-3.43113669
White noise	0

C. Augmented Dickey–Fuller Test

Given a nonstationary time series processed by an SG filter, we need to check whether it is stationary by using differencing operations; otherwise, ARIMA cannot be applied to it. We adopt the augmented Dickey–Fuller (ADF) test [35] to do so. It judges whether a given autoregressive process is stationary by determining if there is a unit root in a time series. If there is no unit root, the time series is stationary. Given (2), the AR process with an order of p is given as

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} \tag{6}$$

where ϕ_i $(i \in \{1, 2, \ldots, p\})$ denotes a coefficient in the AR process, which is determined with the least-squares method.

Its characteristics equation is obtained as

$$\lambda^p - \phi_1 \lambda^{p-1} - \cdots - \phi_p = 0 \tag{7}$$

where λ denotes an independent variable.

The workload is stationary if all of its eigenvalues fall into a unit circle. However, the workload is nonstationary if a characteristic root is 1 or larger.

Then, we perform the ADF test with the null assumption of $\rho = 0$ and the other one of $\rho < 0$. ρ is obtained as

$$\rho = \phi_1 + \phi_2 + \cdots + \phi_p - 1. \tag{8}$$

π means the value of the test statistic. It is compared with its corresponding critical value in the ADF test. If π is smaller, the null assumption is not accepted. Specifically, π is given as

$$\pi = \frac{\widehat{\rho}}{\mathrm{S}(\widehat{\rho})} \tag{9}$$

where $\mathrm{S}(\widehat{\rho})$ denotes the standard deviation of $\widehat{\rho}$.

If the current time series does not pass the ADF test, its difference is further calculated and the ADF test is conducted repeatedly. Here, the ADF level is set to 1%, 5%, and 10%, respectively. They mean the rejection degree of the original assumption. For example, if an ADF value is smaller than the critical value of 1%, the time series is stationary; otherwise, it is not. Here, the significant level is 0.01 or 0.05, and we need the differencing if p is larger than it. We then adopt the test of ADF unit root to analyze the workload in the Google cluster. Table II shows the result of the ADF test.

The obtained value of ADF is further compared with critical values to check whether the workload is stationary. It is observed that the ADF value is smaller than the critical value of 1%, and the order of p is also much smaller than the significant level of 0.01. It demonstrates that the nonstationary time series can be stationary with differencing, and therefore, ARIMA can be further applied to its stationary version.

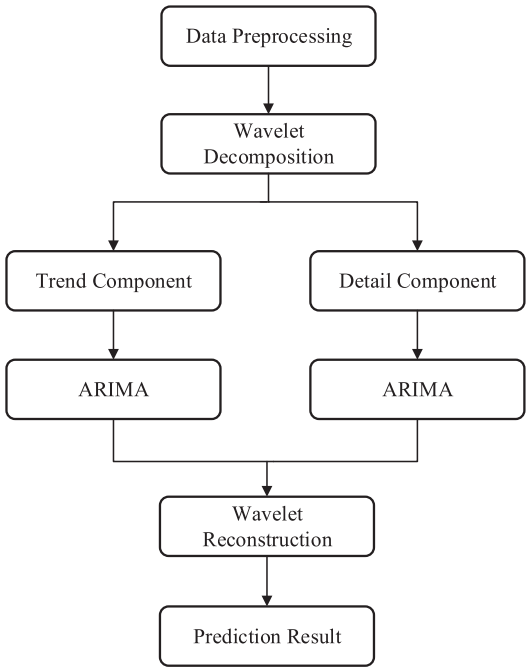


Fig. 5. Framework of the prediction model.

V. PREDICTION MODELS OF WORKLOAD

This section introduces the details of prediction models for workload. We first preprocess the data with the method introduced in Section IV. Then, the noise in the workload time series is removed. We further decompose it into two splits with the wavelet decomposition. One split includes the trend information of the original time series, and the other one the detail information. Then, these two splits are further smoothed with an SG filter. Following this step, two time series are obtained to contain different characteristics. We further adopt ARIMA to build two prediction models for these two time series. Then, we can obtain the trend and detail values in the next time interval, respectively. At last, we obtain the number of tasks to arrive in the next time interval with wavelet reconstruction. Fig. 5 shows the framework of the proposed prediction model.

A. Haar Wavelet Decomposition

Wavelet decomposition is effective to process nonlinear and nonstationary time series because it removes the nonstationary data from a time series and increases the prediction accuracy. Wavelets (e.g., Daubechies wavelets, Mexican Hat wavelets, and Morlet wavelets) can be used to obtain the detail information at multiple resolution scales. However, these wavelets are slower than Haar wavelets. Haar wavelets only need additions and include many 0's as their elements. Therefore, their execution time is very short. Besides, they can discover local information about the time series. Therefore, we first adopt the Haar wavelet decomposition [36] to investigate information of the time series with different scales of resolution. For a time series with 2^n numbers, we arrange every two successive data into a group. It is further transformed into a new time series with 2^{n-1} groups. We then obtain

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

the sum and difference of any two successive data in each group and produce two new time series. This step is named one-stage wavelet transformation. It is recursively repeated to obtain the final 2^{n-1} differences and sum. Here, we adopt the Haar wavelet transformation, which is a typical type of wavelet transformation. The Haar mother wavelet is

$$\Theta(\gamma) = \begin{cases} 1, & 0 \le \gamma \le 1/2 \\ -1, & 1/2 \le \gamma \le 1 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where γ denotes the support domain of Haar wavelet.

Then, we have the following scaling equation, i.e.,

$$\theta(\gamma) = \begin{cases} 1, & 0 \le \gamma \le 1 \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Here, we adopt the Haar matrix Γ, which is shown as

$$\Gamma = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{12}$$

Any time series $(x_1, x_2, x_3, x_4, \ldots, x_{2t}, x_{2t+1})$ of the same length can be transformed into a series of vectors $((x_1, x_2), (x_3, x_4), \ldots, (x_{2t}, x_{2t+1}))$ by using a Haar matrix. Then, we calculate the sum (s) and difference (d) of each vector, and record them as $((s_1, d_1), (s_2, d_2), \ldots, (s_t, d_t))$, which is transformed with the Haar wavelet operation.

One-stage Haar wavelet decomposition is next performed on the workload, and two new time series are obtained to show several features of the original workload. The first time series shows the changing trend in the original workload. In the second time series, the detail information is adopted to obtain the changes between any two successive observations in the original workload. In this way, trends and details can be further adopted to increase the prediction accuracy of the proposed model.

B. Symlet Wavelet Decomposition

A Symlet wavelet function [37] is an approximate symmetric one, and an improved variant of the Daubechies function. A valid range of a Symlet wavelet is $2Z - 1$ where Z is a vanishing moment. Symlet wavelet has better regularity and there is no stationary expression or wavelet base. Its discrete expression is $m_0(\alpha) = (1/\sqrt{2}) \sum_{v=0}^{2Z-1} h_v e^{-iv\alpha}$, and its modulus square is $|m_0(\alpha)|^2 = [\cos^2(\alpha/2)]^Z P[\sin^2(\alpha/2)]$ where $P(y) = \sum_{v=0}^{2Z-1} C_v^{Z-1+v} y^v$. In addition, α is an angular vector, and h_v is a given parameter. For clarity, $W = |m_0(\alpha)|^2$ and $U = \mathrm{e}^{i\alpha}$. Then, W can be decomposed into $W(z) = Q(U)Q(1/U)$. Here, U and $(1/U)$ are pairs of roots, and $Q(\cdot)$ denotes a decomposed function.

Symlet wavelet is the same as Daubechies wavelet [38] with respect to continuity, support length and filter length, but has better symmetry. Therefore, it can be used to decompose and reconstruct a signal without phase distortion to a certain extent. In this work, one-stage Sym4 wavelet decomposition is performed on workload, and two new time series are obtained to show several features of the original workload. The first time series shows the changing trend information in the original workload. In the second time series, the detail information is adopted to obtain changes between any two successive observations in the original workload. In this way, the trend and detail information can be further adopted to increase the prediction accuracy of the proposed method.

C. ARIMA

Following earlier studies [22], Chehelgerdi-Samani and Safi-Esfahani [17] proposed a novel method to obtain ARIMA. It contains three steps including diagnostic checking, parameter estimation, and model identification. In model identification, a sequence is produced by ARIMA, and it has several autocorrelation properties. By comparing theoretical patterns and empirical ones, we might obtain several candidate models for a given workload. Box and Jenkins adopt a partial autocorrelation function (PACF) and ACF of the given data to determine the order of ARIMA. Then, the data in the future is predicted based on present and past data in the sequence, and therefore, it is commonly adopted to realize short-term prediction.

The resulting model is expressed with ARIMA(p, d, q) where p, q, and d denote the order of the AR model, the order of the MA model, and the differencing degree, respectively. ARIMA can be viewed as the combination of AR(p) and MA(q). Note that AR(p) is the same as (6) of an AR process with the order of p in Section IV-C. We adopt MA(q) to process white noise in workload. Then, x_t is obtained as

$$x_t = \varepsilon_t + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \cdots + \beta_q \varepsilon_{t-q}. \tag{13}$$

In ARIMA, the predicted value of each variable is a linear combination of some random errors and past observations. Specifically

$$x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + \varepsilon_t + \beta_1 \varepsilon_{t-1} + \cdots + \beta_q \varepsilon_{t-q}. \tag{14}$$

In (14), ε_t and x_t denote a random error and an actual value at time interval t, respectively. β_j $(j \in \{1, 2, \ldots, q\})$ is a coefficient of MA(q). We assume that random errors are identically and independently distributed around the mean of zero.

AIC (A_{IC}) [39] is a criterion to measure the quality of a model. ARIMA includes an AR model with the order of p and an MA model with the order of q. Following Akaike's theory, the number of parameters in ARIMA is $p + q$. A_{IC} is given as

$$A_{IC} = 2k - 2\ln(\widehat{L}) \tag{15}$$

where k denotes the number of parameters, and $\widehat{L}$ denotes the maximum-likelihood function value of the prediction model.

D. Training Steps

The training stage of our model is described as follows. Given a nonstationary workload, p and q are affected by the number of peaks in ACF and PACF. Consequently, p and q cannot be determined by them. Then, we specify orders of AR and MA through many experiments in Section VI-A. It includes the following steps:

1) specify orders of AR and MA models;

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

2) construct the ARIMA model with p and q, and obtain its A_{IC};
3) select the ARIMA model with the smallest A_{IC}.

After the above operations, the prediction model is obtained for the original workload. We can easily realize the proposed prediction in realistic CDCs if the workload is available in each time slot. Here, the SG filter is adopted to smooth workload and decompose it into detail and trend components with wavelet decomposition. Then, ARIMA is applied to the two components, and their predicted results are further reconstructed with the wavelet reduction method. In this way, the predicted number of tasks arriving in the next time slot is obtained.

E. Wavelet Reconstruction and Prediction Results

The testing stage of our model is described as follows. At each time slot k, the data from time slots $k-1$ to $k-q$ are obtained. The data are first preprocessed and decomposed by the wavelets to obtain the trend result [trend(k)] and the detail one [detail(k)] in time slot k. The trained ARIMA model is used for the prediction results, which are further processed by wavelet reconstruction. The wavelet reconstruction process is given as

$$\hat{y}_k = \text{trend}(k) + \text{detail}(k) \tag{16}$$

where $\hat{y}_k$ denotes the prediction result in time slot k.

Finally, the prediction result $\hat{y}_k$ in time slot k is obtained. The above steps are repeated in each time slot.

VI. Performance Evaluation

A. Data Preprocessing

This work chooses the workload in the first 20 days as the training data. We choose the workload from the 21st to 25th days as the test data and select the workload in the last four days as the testing data. Fig. 6 illustrates changes of variance and mean of the Google workload trace 2011. It is obvious that the mean value changes greatly and the variance varies between 0 and 1.5. Besides, they do not keep stable around some constant, and therefore, this workload time series fails to meet the stationarity constraint. To reduce the noise in the data, this work adopts an SG filter. Fig. 7 illustrates the original and smoothed data of the Google workload trace 2011. It is worth noting that the aggregated task arriving rates are transformed with the logarithm function. It is shown that the data smoothed by the SG filter is smoother than the original one.

However, the operation of smoothing may eliminate the valuable information in the original workload. Therefore, the following experiments are conducted to demonstrate its effectiveness. We adopt four types of workloads including the original sequence, and sequences smoothed by SG, MA, and MM filters, respectively. As shown in Table III, the window sizes in them are set to 5 because the best filtering results are obtained after multiple trials.

A separate prediction model is obtained for each of the four sequences. The prediction accuracy is measured with the

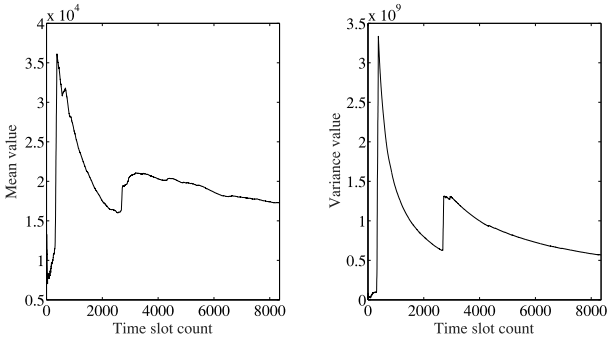


Fig. 6. Mean and variance changes of the Google workload trace 2011.

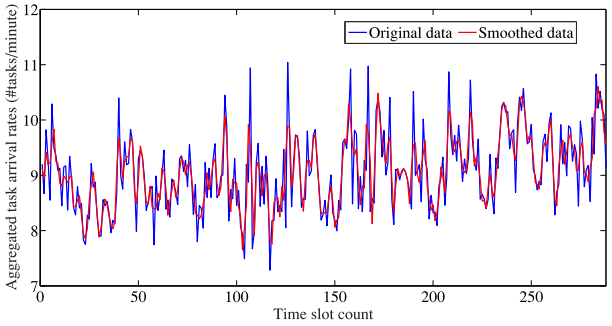


Fig. 7. Original and smoothed data of the Google workload trace 2011.

TABLE III
M_{SE} Under Varying Window Sizes

Size \ M_{SE}	MM Filter	MA Filter	SG Filter
3	0.3694	0.3856	0.4320
5	**0.2046**	**0.2158**	**0.0704**
7	0.2504	0.3410	0.1685
9	0.3946	0.3645	0.1728

TABLE IV
M_{SE} of Four Workloads

Group \ M_{SE}	Dataset A	Dataset B	Dataset C
Original data	0.3795	0.3757	0.8478
Data filtered with MA	0.1974	0.2080	0.3990
Data filtered with MM	0.1354	0.1431	0.2668
Data filtered with SG	**0.0704**	**0.0810**	**0.1694**

mean-squared error (M_{SE}) [40], [41], i.e.,

$$M_{SE} = \frac{\sum_{\omega=1}^{\Omega}(\widehat{y}_\omega - y_\omega)^2}{\Omega}. \tag{17}$$

In (17), y_ω and $\widehat{y}_\omega$ denote the original and predicted data, respectively, at time slot ω, and Ω denotes the total number of observations. It is worth noting that a smaller value of M_{SE} means higher prediction accuracy. This work adopts three datasets marked as A, B and C, respectively. Table IV shows M_{SE} of four sequences corresponding to them. It is observed that the SG filter achieves the highest prediction accuracy.

Besides, as discussed in Section V-C, parameters, including d, p, and q, need to be set in ARIMA. d is determined in the smoothing operation. To determine p and q, this work investigates the correlation by calculating ACF and PACF of

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

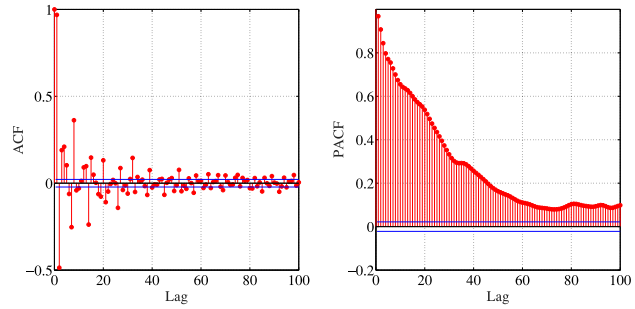


Fig. 8. ACF and PACF of a sequence of the Google workload trace 2011.

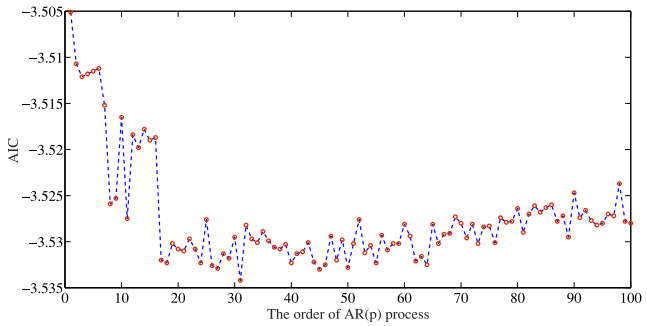


Fig. 9. A_{IC} of ARIMA with varying p.

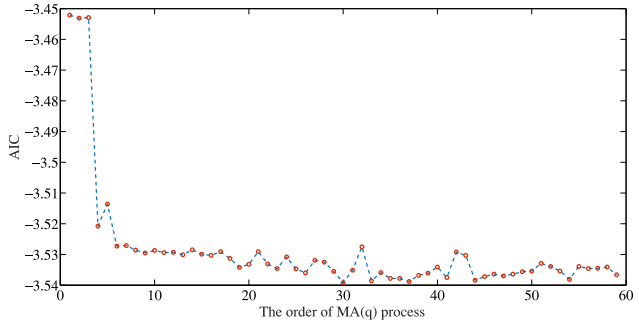


Fig. 10. A_{IC} of ARIMA with varying q.

the workload sequence with the Box–Jenkins approach in the training time. Fig. 8 illustrates the results.

Fig. 8 shows that the lags of ACF and PACF for a sequence of the Google workload trace 2011 are unable to cut off in black lines. The curves of ACF and PACF fail to converge to their specified thresholds, and therefore, exact values of p and q cannot be obtained based on ACF and PACF. Then, A_{IC} is calculated to specify the best setting of p and q, which provides the most accurate prediction model. Specifically, p and q are limited to given ranges, and each ARIMA model can have multiple combinations of parameter settings. Then, this work determines the best model with the highest prediction performance by using A_{IC}. Fig. 9 shows the variations of A_{IC} of ARIMA with different values of p, and A_{IC} is the least if $p = 31$. Fig. 10 shows variations of A_{IC} of ARIMA with different values of q. A_{IC} is the least if $q = 30$.

Figs. 9 and 10 show that the best ARIMA model with the highest prediction accuracy is obtained when $p = 31$ and $q = 30$. Table V shows the comparison of prediction results of multiple ARIMA models with different combinations of

TABLE V
RESULTS OF ARIMA MODELS WITH DIFFERENT PARAMETER SETTINGS

p-value, q-value	M_{SE}	A_{IC}
(31, 30)	**0.0595**	**-3.5355**
(31, 15)	0.0602	-3.5285
(31, 0)	0.0603	-3.4506
(20, 30)	0.0596	-3.5320
(10, 30)	0.0621	-3.5261
(5, 30)	0.0646	-3.5269

TABLE VI
M_{SE} OF THREE WAVELET DECOMPOSITION METHODS

Group \ M_{SE}	Dataset A	Dataset B	Dataset C
Original data	0.5484	0.6378	0.3611
Data with Haar	0.2813	0.4435	0.2295
Data with Daubechies	0.2059	0.2115	0.1293
Data with Symlet	**0.1541**	**0.1349**	**0.0865**

p and q. The prediction result is evaluated in terms of M_{SE}. Table V shows the best prediction is achieved when A_{IC} is the least. In addition, the ARIMA model with the best parameter setting is determined with the above experiments.

B. Evaluation Criteria

This work adopts three metrics, including M_{SE} [40], [41], R^2 [42], and the mean absolute percentage error (M_{APE}) [43], [44] to evaluate the prediction accuracy. R^2 measures the goodness-of-fit of a given prediction model, i.e.,

$$R^2 = \frac{\sum_{\omega=1}^{\Omega}(\widehat{y}_\omega - \bar{y})^2}{\sum_{\omega=1}^{\Omega}(y_\omega - \bar{y})^2}. \tag{18}$$

In (18), $\bar{y}$ denotes the real-life output, and $\bar{y} = (1/\Omega)\sum_{\omega=1}^{\Omega} y_\omega$. Besides, $\widehat{y}_\omega$ denotes the predicted result, and Ω denotes the number of records. R^2 shows the fitting ability of a prediction model, and it is within the range of [0, 1]. $R^2 = 1.0$ means a perfect match between the predicted data and the actual one. In addition, for a prediction model, lower M_{APE} means higher fitting ability where

$$M_{\text{APE}} = \frac{1}{\Omega}\sum_{\omega=1}^{\Omega}\frac{|\widehat{y}_\omega - y_\omega|}{y_\omega}. \tag{19}$$

C. Prediction Result

This work adopts wavelet decomposition to discover more characteristics of the sequence of the Google cluster workload trace 2011. This work adopts three types of wavelet decomposition methods, including Haar, Daubechies, and Symlet wavelets. As shown in Table VI, Symlet is the best one after multiple trials. Then, the trend and detail components with the same length of 2883 are obtained. Trend and detail components on the first day of our training set are shown in Fig. 11. Here, the order of Symlet wavelet is 4. Wavelet decomposition can be used to obtain multilevel outputs for multiscale variations. By applying it to a nonstationary time series, we analyze its low and high frequencies, from which we find its global information and the information of hidden patterns. By applying the Symlet wavelet to the original time series, the feature information is effectively extracted from its trend and detail

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

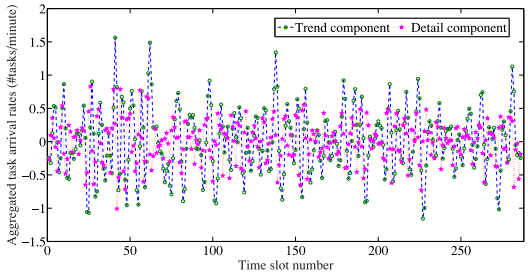


Fig. 11. Result with the wavelet decomposition for Google cluster workload trace 2011.

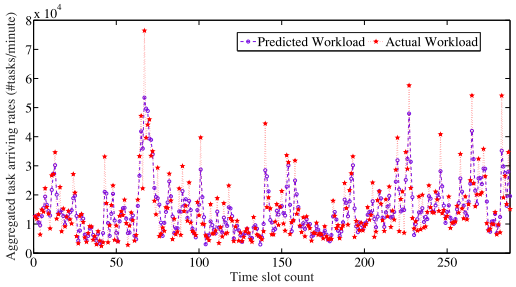


Fig. 12. Predicted and actual workloads of all three types in Google cluster workload trace 2011.

sequences. At the same time, there are almost no extreme points in the decomposed sequences, which tends to affect the prediction accuracy significantly. Hence, the modeling ability and prediction accuracy are further improved.

Wavelet decomposition produces two subsequences, including detail and trend components, respectively. We apply ARIMA to them and obtain their predicted results, which are further processed by the wavelet reconstruction to provide the future number of tasks arriving in the next time interval. Besides, we apply the proposed prediction model to predict seven different workload series from two different real-life traces, i.e., Google cluster workload traces 2011 and 2019, to demonstrate its effectiveness. Fig. 12 shows the predicted and actual total workloads while Fig. 13 illustrates those of each type in Google cluster workload trace 2011. Fig. 14 shows the predicted and actual total workloads while Fig. 15 illustrates those of each type in Google cluster workload trace 2019. It is clear that the proposed prediction model achieves high-accuracy prediction for the total workload and workload of each type.

To demonstrate the performance of SWARIMA, we compare it with SVM [45], BPNN [18], recurrent neural network (RNN) [46], LSTM [19], and ARIMA [11] on the same workloads. Their prediction accuracy is evaluated in terms of M_{SE}, R^2, and M_{APE}. Here, BPNN includes input, hidden, and output layers. BPNN adopts a gradient descent method to iteratively change its network parameters in its training process. BPNN has good nonlinear modeling ability, and well fits different kinds of data. To obtain better performance of the training process of BPNN, this work compares the prediction performance of BPNN with different numbers of hidden nodes. Table VII shows that the least M_{SE} is achieved if there are 20 hidden nodes in BPNN.

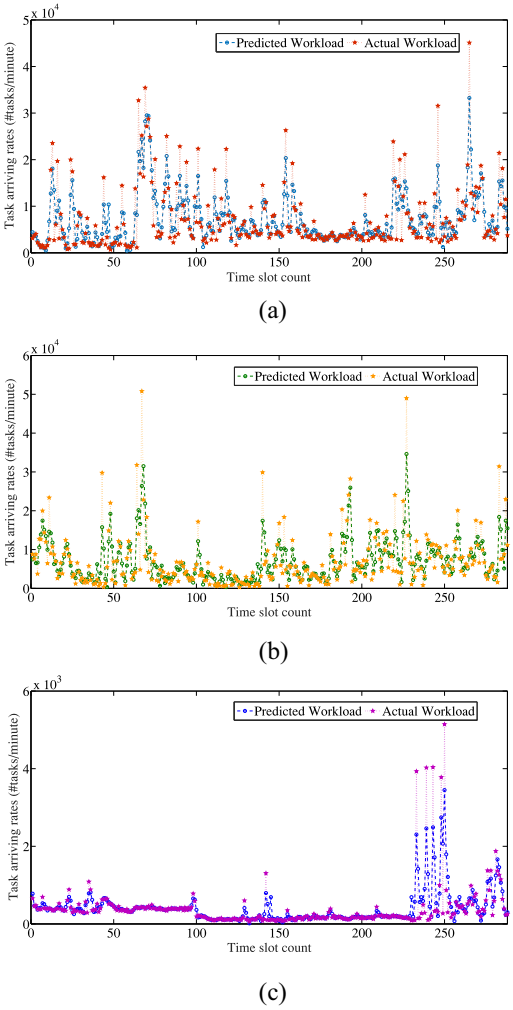


Fig. 13. Predicted workloads of types 1–3 in Google cluster workload trace 2011, respectively. (a) Type 1. (b) Type 2. (c) Type 3.

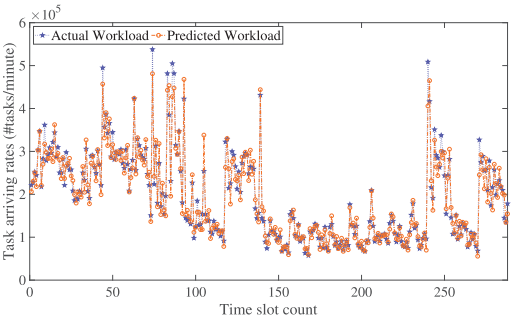


Fig. 14. Predicted and actual workloads of all four types in Google cluster workload trace 2019.

Tables VIII and IX show M_{SE}, R^2, M_{APE}, and the training time of different prediction methods given the same data from Google cluster workload traces 2011 and 2019. It is shown that the prediction accuracy of SWARIMA is better than that of other methods [11], [18], [19], [45], [46] with respect to the above four metrics. The time series decomposed by wavelet decomposition methods provides more features about the original time series. The SG filter and wavelet decomposition smooth the original workload and improve the prediction

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

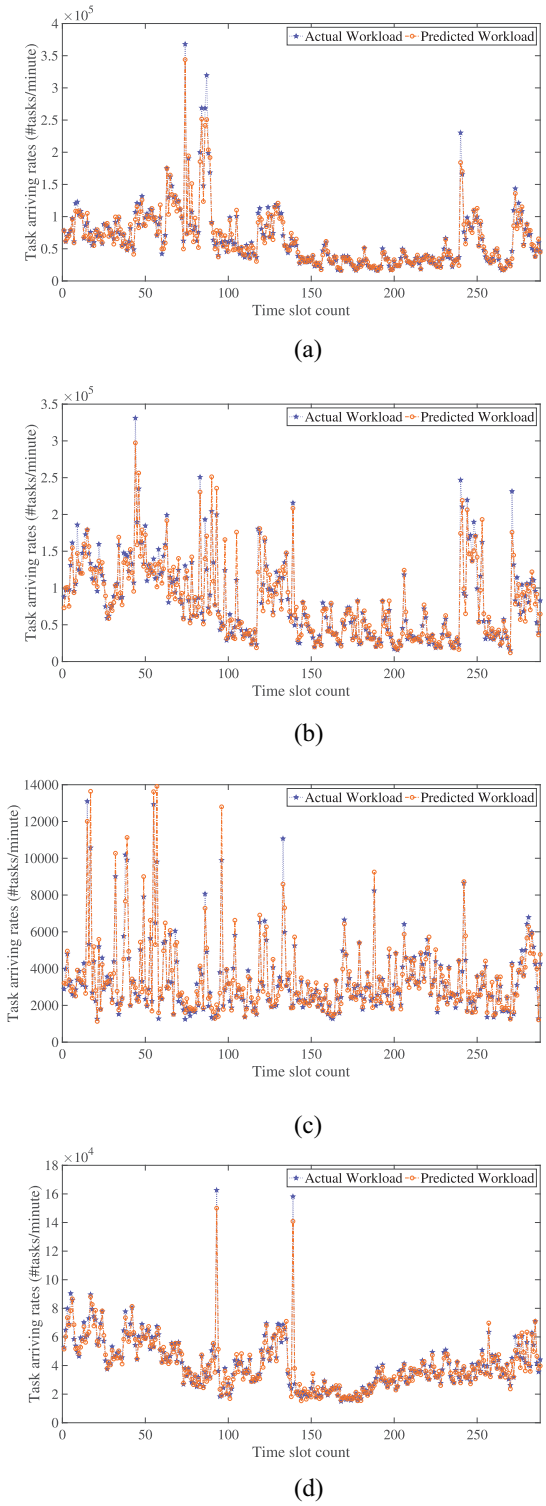


Fig. 15. Predicted workloads of types 1–4 in Google cluster workload trace 2019, respectively. (a) Type 1. (b) Type 2. (c) Type 3. (d) Type 4.

TABLE VII
M_{SE} WITH DIFFERENT NUMBERS OF HIDDEN NODES

Number of hidden nodes	M_{SE}	Number of hidden nodes	M_{SE}
10	0.061	50	0.060
20	**0.059**	60	0.061
30	0.062	70	0.062
40	0.060	80	0.060

TABLE VIII
PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS FOR GOOGLE CLUSTER WORKLOAD TRACE 2011

Methods	M_{SE}	R^2	M_{APE}	Training time (sec.)
SVM	0.078	0.085	0.674	51.54
ARIMA	0.066	0.066	0.664	62.44
BPNN	0.084	0.195	0.685	26.96
RNN	0.071	0.185	0.694	66.54
LSTM	0.068	0.201	0.673	51.38
WSVM	0.075	0.197	0.643	105.53
WARIMA	0.069	0.382	0.632	91.10
WBPNN	0.057	0.434	0.678	183.52
WRNN	0.052	0.452	0.664	99.25
WLSTM	0.047	0.482	0.676	105.32
SSVM	0.074	0.170	0.625	52.73
SARIMA	0.066	0.474	0.647	97.55
SBPNN	0.045	0.543	0.645	178.56
SRNN	0.067	0.492	0.644	69.25
SLSTM	0.052	0.571	0.637	48.52
SWSVM	0.035	0.618	0.524	123.73
SWARIMA	**0.017**	**0.759**	**0.386**	**94.33**
SWBPNN	0.023	0.671	0.465	172.89
SWRNN	0.021	0.687	0.454	120.86
SWLSTM	0.019	0.704	0.404	135.21

TABLE IX
PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS FOR GOOGLE CLUSTER WORKLOAD TRACE 2019

Methods	M_{SE}	R^2	M_{APE}	Training time (sec.)
SVM	0.147	0.391	0.024	37.37
ARIMA	0.143	0.405	0.024	40.47
BPNN	0.080	0.665	0.018	28.80
RNN	0.077	0.678	0.017	48.92
LSTM	0.116	0.540	0.020	53.05
WSVM	0.044	0.818	0.013	91.35
WARIMA	0.020	0.916	0.009	88.41
WBPNN	0.025	0.898	0.010	90.07
WRNN	0.044	0.817	0.013	92.33
WLSTM	0.032	0.854	0.012	164.26
SSVM	0.120	0.577	0.021	42.82
SARIMA	0.035	0.840	0.013	79.51
SBPNN	0.044	0.816	0.013	97.53
SRNN	0.060	0.756	0.015	85.34
SLSTM	0.049	0.793	0.014	99.77
SWSVM	0.032	0.865	0.011	109.70
SWARIMA	**0.010**	**0.952**	**0.006**	**90.56**
SWBPNN	0.023	0.903	0.010	113.10
SWRNN	0.026	0.893	0.010	130.72
SWLSTM	0.026	0.890	0.010	173.11

accuracy of SWARIMA. The wavelet decomposition also increases the prediction accuracy because Symlets provide symmetric wavelets, and they have the least asymmetry and the largest number of vanishing moments for a given compact support. The Symlet wavelet transformation decomposes the original sequence into multilevel outputs. Therefore, the SG filter and wavelet decomposition together greatly reduce noise in the original sequence, thereby improving the prediction accuracy. Compared with other machine learning methods, ARIMA achieves significant improvement in prediction accuracy and training time after using SG filtering and wavelet decomposition. For example, in Table VIII, MSE, MAPE, and the training time of SWARIMA for Google cluster workload trace 2011 are all the smallest among all SW-related models, which are 0.017, 0.386, and 94.33, respectively. In Table IX, these values for Google cluster workload trace 2019 are 0.010,

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

0.952, and 90.56, respectively, which are the least. In addition, its R^2 of 0.759 is the largest. The reason is that the data processed by SG filtering and wavelet decomposition has lower noise and clearer data characteristics, and ARIMA is built with it in an easier manner.

VII. Conclusion

Current CDCs consume a great amount of energy to provide services for users around the world. Their energy consumption and efficiency become critically important. The prediction accuracy of future incoming tasks is vital to realize accurate resource provisioning for user tasks. However, the irregularity and complexity of workload make it challenging to achieve it. This work designs a novel prediction approach named SWARIMA by combining SG filter in wavelet decomposition, and ARIMA. Experimental results show that its prediction accuracy and the learning speed are better than its state-of-the-art peers.

It is worth noting that some parameters used in this work are only designed for Google cluster traces in 2011 and 2019. Their setting needs to be tuned if SWARIMA is applied to other datasets. In addition, SWARIMA is designed for non-stationary sequences. Thus, its prediction performance might not be good enough if sequences are already stationary. Our future work is to extend SWARIMA with some new deep learning algorithms to yield better performance. Besides, it is interesting to investigate temporal and spatial features in tasks, and use them to boost accuracy.

References

[1] H. Yuan, J. Bi, J. Zhang, and M. Zhou, "Energy consumption and performance optimized task scheduling in distributed data centers," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 9, pp. 5506–5517, Sep. 2022.

[2] D. Singh, K. Dwarakanath, and R. Pasumarthy, "Event-triggered control design for systems with exogenous inputs: Application for auto-scaling of cloud-hosted Web servers," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 8, pp. 5201–5211, Aug. 2022.

[3] J. Bi, H. Yuan, J. Zhang, and M. Zhou, "Green energy forecast-based bi-objective scheduling of tasks across distributed clouds," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 3, pp. 619–630, Jul.–Sep. 2022.

[4] D. Saxena, J. Kumar, A. K. Singh, and S. Schmid, "Performance analysis of machine learning centered workload prediction models for cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 4, pp. 1313–1330, Apr. 2023.

[5] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Forecasting cloud application workloads with cloudinsight for predictive resource management," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1848–1863, Sep. 2022.

[6] D. N. Ganapathy and K. P. Joshi, "A semantically rich framework to automate cloud service level agreements," *IEEE Trans. Services Comput.*, vol. 16, no. 1, pp. 53–64, Jan./Feb. 2023.

[7] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu, and H. Tenhunen, "Energy-aware VM consolidation in cloud data centers using utilization prediction model," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 524–536, Apr.–Jun. 2019.

[8] D. Saxena, A. K. Singh, and R. Buyya, "OP-MLB: An online VM prediction-based multiobjective load balancing framework for resource management at cloud data center," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2804–2816, Oct.–Dec. 2022.

[9] Y. Bao, Y. Peng, and C. Wu, "Deep learning-based job placement in distributed machine learning clusters with heterogeneous workloads," *IEEE/ACM Trans. Netw.*, vol. 31, no. 2, pp. 634–647, Apr. 2023.

[10] A. K. Singh, D. Saxena, J. Kumar, and V. Gupta, "A quantum approach towards the adaptive prediction of cloud workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 12, pp. 2893–2905, Dec. 2021.

[11] X. Fu and C. Zhou, "Predicted affinity based virtual machine placement in cloud computing environments," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 246–255, Mar. 2020.

[12] Y. Fanjiang, Y. Syu, and W. Huang, "Time series QoS forecasting for Web services using multi-predictor-based genetic programming," *IEEE Trans. Services Comput.*, vol. 15, no. 3, pp. 1423–1435, Jun. 2022.

[13] B. Feng, Z. Ding, and C. Jiang, "FAST: A forecasting model with adaptive sliding window and time locality integration for dynamic cloud workloads," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1184–1197, Mar./Apr. 2023.

[14] Z. Ding, B. Feng, and C. Jiang, "COIN: A container workload prediction model focusing on common and individual changes in workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4738–4751, Dec. 2022.

[15] M. Zakarya et al., "Epcaware: A game-based, energy, performance and cost-efficient resource management technique for multi-access edge computing," *IEEE Trans. Services Comput.*, vol. 15, no. 3, pp. 1634–1648, Jun. 2022.

[16] J. Bi, X. Zhang, H. Yuan, J. Zhang, and M. Zhou, "A hybrid prediction method for realistic network traffic with temporal convolutional network and LSTM," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 1869–1879, Jul. 2022.

[17] M. Chehelgerdi-Samani and F. Safi-Esfahani, "PCVM.ARIMA: Predictive consolidation of virtual machines applying ARIMA method," *J. Supercomput.*, vol. 77, no. 3, pp. 2172–2206, Mar. 2021.

[18] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Gener. Comput. Syst.*, vol. 81, pp. 41–52, Apr. 2018.

[19] S. Tuli, S. S. Gill, P. Garraghan, R. Buyya, G. Casale, and N. R. Jennings, "START: Straggler prediction and mitigation for cloud computing environments using encoder LSTM networks," *IEEE Trans. Services Comput.*, vol. 16, no. 1, pp. 615–627, Jan./Feb. 2023.

[20] O. Runsewe and N. Samaan, "Cloud resource scaling for time-bounded and unbounded big data streaming applications," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 504–517, Apr.–Jun. 2021.

[21] D. Huang et al., "Harnessing data movement in virtual clusters for in-situ execution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 3, pp. 615–629, Mar. 2019.

[22] Y. Xie et al., "Real-Time prediction of docker container resource load based on a hybrid model of ARIMA and triple exponential smoothing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 1386–1401, Apr.–Jun. 2022.

[23] Q. Sun, Z. Tan, and X. Zhou, "Workload prediction of cloud computing based on SVM and BP neural networks," *J. Intell. Fuzzy Syst.*, vol. 39, no. 3, pp. 2861–2867, Oct. 2020.

[24] X. Hou, K. Wang, C. Zhong, and Z. Wei, "ST-Trader: A spatial-temporal deep neural network for modeling stock market movement," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 5, pp. 1015–1024, May 2021.

[25] Y. Zhang, Y. Zhou, H. Lu, and H. Fujita, "Traffic network flow prediction using parallel training for deep convolutional neural networks on spark cloud," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7369–7380, Dec. 2020.

[26] S. Gupta, A. D. Dileep, and T. A. Gonsalves, "Online sparse BLSTM models for resource usage prediction in cloud datacentres," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2335–2349, Dec. 2020.

[27] G. J. Portella, E. Nakano, G. N. Rodrigues, A. Boukerche, and A. C. M. A. Melo, "A novel statistical and neural network combined approach for the cloud spot market," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 278–290, Jan.–Mar. 2023.

[28] L. Fang, G. Shen, H. Luo, C. Chen, and Z. Zhao, "Automatic extraction of roadside traffic facilities from mobile laser scanning point clouds based on deep belief network," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 1964–1980, Apr. 2021.

[29] L. Zhang et al., "A novel hybrid model for docker container workload prediction," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 2726–2743, Sep. 2023.

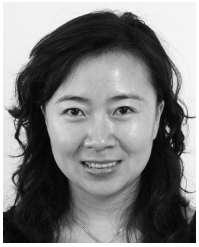
[30] Y. Lu, L. Liu, J. Panneerselvam, X. Zhai, X. Sun, and N. Antonopoulos, "Latency-based analytic approach to forecast cloud workload trend for sustainable datacenters," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 3, pp. 308–318, Jul.–Sep. 2020.

[31] O. A. Abdul-Rahman and K. Aida, "Google users as sequences: A robust hierarchical cluster analysis study," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 167–179, Jan.–Mar. 2020.

[32] H. Sami, A. Mourad, H. Otrok, and J. Bentahar, "Demand-driven deep reinforcement learning for scalable fog and service placement," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2671–2684, Sep.–Oct. 2022.

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.

[33] J. Bi, H. Ma, H. Yuan, and J. Zhang, "Accurate prediction of workloads and resources with multi-head attention and hybrid LSTM for cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 3, pp. 375–384, Jul./Sep. 2023.
[34] A. Maalouf, I. Jubran, and D. Feldman, "Fast and accurate least-mean-squares solvers for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9977–9994, Dec. 2022.
[35] J. Bi, H. Yuan, L. Zhang, and J. Zhang, "SGW-SCN: An integrated machine learning approach for workload forecasting in geo-distributed cloud data centers," *Inf. Sci.*, vol. 481, pp. 57–68, May 2019.
[36] E. Wu, G. Zhou, L. Zhu, C. Wei, H. Ren, and R. Sheng, "Rotated sphere haar wavelet and deep contractive auto-encoder network with fuzzy gaussian SVM for pilot's pupil center detection," *IEEE Trans. Cybern.*, vol. 51, no. 1, pp. 332–345, Jan. 2021.
[37] S. Ataky and A. Koerich, "Multiresolution texture analysis of histopathologic images using ecological diversity measures," *Expert Syst. Appl.*, vol. 224, Apr. 2023, Art. no. 119972.
[38] C. Napoli, G. Magistris, C. Ciancarelli, F. Corallo, F. Russo, and D. Nardi, "Exploiting wavelet recurrent neural networks for satellite telemetry data modeling, prediction and control," *Expert Syst. Appl.*, vol. 206, Nov. 2022, Art. no. 117831.
[39] H. Zheng, et al., "Bi-CCD: Improved continuous change detection by combining forward and reverse change detection procedure," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, Jul. 2022.
[40] E. K. P. Chong, "Well-conditioned linear minimum mean square error estimation," *IEEE Control Syst. Lett.*, vol. 6, pp. 2431–2436, 2022.
[41] Z. Wang, M. Badiu, and J. Coon, "On the value of information and mean squared error for noisy gaussian models," *IEEE Commun. Lett.*, vol. 26, no. 9, pp. 2023–2026, Sep. 2022.
[42] G. Huang, J. Benesty, I. Cohen, and J. Chen, "Kronecker product multichannel linear filtering for adaptive weighted prediction error-based speech dereverberation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 1277–1289, Mar. 2022.
[43] Y. Liu, Q. Zhang, and Z. Lv, "Real-time intelligent automatic transportation safety based on big data management," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9702–9711, Jul. 2022.
[44] B.-F. Wu, Y.-C. Wu, and Y.-W. Chou, "A compensation network with error mapping for robust remote photoplethysmography in noise-heavy conditions," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–11, Jan. 2022.
[45] W. Wang, X. Du, D. Shan, R. Qin, and N. Wang, "Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1634–1646, Jul.–Sep. 2022.
[46] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 3, pp. 940–954, Mar. 2022.



Jing Bi (Senior Member, IEEE) received the Ph.D. degree in computer application technology from Northeastern University, Shenyang, China, in 2011.

She is currently an Associate Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. She has over 80 publications, including journal and conference papers. Her research interests include distributed computing, cloud computing, large-scale data analysis, machine learning, and performance optimization.

Dr. Bi was the recipient of the IBM Fellowship Award and the Best Paper Award-Finalist in the 16th IEEE International Conference on Networking, Sensing and Control.

Haitao Yuan (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the New Jersey Institute of Technology (NJIT), Newark, NJ, USA, in 2020.

He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms.

Dr. Yuan received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC.

Shuang Li received the B.S. degree in software engineering from the Beijing University of Technology, Beijing, China, in 2018, where he is currently pursuing the master's degree in software engineering from the Faculty of Information Technology, School of Software Engineering.

His research interests include cloud computing, data center, energy management, big data, time-series prediction, machine learning, and deep learning.

Mr. Li was the recipient of the Best Paper Award-Finalist in the 16th IEEE International Conference on Networking, Sensing and Control.

Kaiyi Zhang received the B.S. degree in software engineering from the Beijing University of Technology, Beijing, China, in 2020, where he is currently pursuing the master's degree in software engineering from the Faculty of Information Technology, School of Software Engineering.

His research interests include cloud computing, data center, energy management, big data, time-series prediction, machine learning, and deep learning.

Jia Zhang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA.

She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair of Engineering and a Professor with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in Earth science.

MengChu Zhou (Fellow, IEEE) received the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He then joined the New Jersey Institute of Technology, Newark, NJ, USA, where he is currently a Distinguished Professor. He has over 900 publications, including 12 books, 600+ journal papers (450+ in IEEE TRANSACTIONS), 28 patents, and 29 book chapters. His interests are in Petri nets, automation, Internet of Things, and big data.

Dr. Zhou is a Fellow of IFAC, AAAS, CAA, and NAI.

Authorized licensed use limited to: SOUTHERN METHODIST UNIV. Downloaded on October 18,2024 at 14:44:31 UTC from IEEE Xplore. Restrictions apply.