IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 15, 2019, accepted March 13, 2019, date of publication March 18, 2019, date of current version April 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2905634

Large-Scale Computing Systems Workload Prediction Using Parallel Improved LSTM Neural Network

XIAOYONG TANG
College of Information Science and Technology, Hunan Agricultural University, Changsha 410128, China
Southern Regional Collaborative Innovation Center for Grain and Oil Crops in China, Hunan Agricultural University, Changsha 410128, China
e-mail: tang_313@163.com

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0204004, in part by the Hunan Provincial Key Research and Development Program under Grant 2018GK2055, in part by the Double First-Class Construction Project of Hunan Agricultural University under Grant SYL201802029.

ABSTRACT In recent years, large-scale computing systems have been widely used as an important part of the computing infrastructure. Resource management based on systems workload prediction is an effective way to improve application efficiency. However, accuracy and real-time functionalities are always the key challenges that perplex the systems workload prediction model. In this paper, we first investigate the dependence on historical workload in large-scale computing systems and build a day and time two-dimensional time-series workload model. We then design a two-dimensional long short-term memory (LSTM) neural network cell structure. Based on this, we propose an improved LSTM prediction model providing its mathematical description and an error back propagation method. Furthermore, to achieve systems resource management real-time requirement, we provide a parallel improved LSTM algorithm that uses a hidden layer week-based dependence and weights parallelization algorithm. The comparative studies, based on the actual workload of the Shanghai Supercomputer Center, demonstrate that our proposed improved LSTM neural network prediction model can achieve higher accuracy and real-time performance in large-scale computing systems.

INDEX TERMS Workload prediction, computing systems, LSTM, neural network, parallel.

I. INTRODUCTION

With the advent of electronic and computing technology, large-scale computing systems have received substantial attention from the business, industry, and academic communities [1], [2]. Supercomputers ranked in the TOP500 list, such as Summit, Sierra, Tianhe-2A [3], have played key roles in large-scale data mining, astronomy, fractal calculations and simulations, civil engineering, seismic data process, and weather prediction. The large-scale computability, multidomain characteristic, and highly dynamic nature of these applications inevitably lead to complex systems resource management [4]. Moreover, the large-scale computing systems consume an ever greater amount of more electrical power and suffer high operational costs. For example, Supercomputers Tianhe-2A has a maximum operating power consumption of 18.482MW [3], which is equivalent to the total energy consumption of 5 universities. These problems have created significant incentive to improve application efficiency in supercomputing systems.

An effective way to deal with these challenges is resource management based on systems workload prediction [5]–[7]. The method can estimate future resource demands in computing systems through identification of historical usage patterns and systems current state. Furthermore, this information on workload is helpful to improve the automatic allocation strategy for resources and save systems operating cost [8], [25]. For low workload, the systems resource management can dynamically power off resources to save energy and costs like network, cooling, and maintenance. The systems can also scale up resources to meet user application requirements as the prediction of workload increases. Therefore, the waste in systems resources and operational cost are minimized while user Service Level Agreements (SLA) and Quality of Service (QoS) are maintained.

The associate editor coordinating the review of this manuscript and approving it for publication was Long Wang.

The most important metric on systems workload prediction models is accuracy, which is evaluated by the difference between predicted results and actual values [9]. Generally, the closer the predicted value is to the actual value, the better the model is. There are many techniques and methods applied in the prediction of computing systems workload, such as, Online Ensemble Learning Approach [10], Auto Regressive and Moving Average models (ARIMAR) [11], Recurrent Neural Network (RNN) [12], Long Short-Term Memory (LSTM) Networks [13]. However, workload prediction is a complex social and economic endeavor involving computing systems characteristics, user problems, and application requirements. Accuracy has always been a difficult and challenging item. There is still room to improve the accuracy of systems workload prediction.

The other important metric for systems workload prediction models is real-time effectiveness. This measures the ability of the systems resource management to carve out enough time to optimize its resources according to future workload prediction, which should be produced in a reasonable and effective time frame [9]. For instance, in 5 minutes, the Supercomputers Tianhe-2A will perform financial Big data analysis and processing systems requiring 12400 computing nodes. The prediction model should complete the computation in a minute. This way, the systems resource management will have sufficient time to prepare these resources. However, regarding current workload prediction models, especially in the context of machine learning [6], [8], [12], [13], [31], most techniques require considerable time for training and learning time, which leads to difficulty in meeting systems resource management real-time demands. One of the effective methods to reduce execution time for prediction models is parallelization technology.

Most workload prediction models, such as ARIMAR [11], RNN [12], LSTM [13], assume that training data is a one-dimensional time series. That is to say, the future systems workload is dependent solely on the preceding one-dimensional information. However, in large-scale computing centers, systems workload has a periodicity in time, date, and season. For each day, the workload may be higher during traditional working hours, and relatively low at other times. There may be similarities in workload across weeks, as well as considerable variations on a seasonable basis. Therefore, we believe that the large-scale computing systems workload has the characteristics of a two-dimensional time series, which is missed by most existing methods.

Motivated by these observations, this paper proposes an improved LSTM neural network prediction method based on a two-dimensional time series workload model. In order to meet real-time requirements, we also design a parallel algorithm. The major contributions of this work are multifold and can be summarized as follows:

- First, this study analyzes the correlation and dependence in historical workload data for large-scale computing systems, and constructs a two-dimensional time series workload model.
- Second, we build a two-dimensional LSTM neural network cell structure and propose an improved LSTM prediction model.
- Third, a parallel improved LSTM algorithm is implemented by using the hidden layer week-based dependence and a weights parallelization algorithm.
- Finally, the evaluation of performance is conducted and experimental results indicate that our proposed Improved LSTM algorithm outperforms traditional LSTM in terms of the prediction accuracy. The parallel improved LSTM can also meet computing systems real-time requirements satisfactorily.

The remainder of the paper is organized as follows: In Section 2, we review related studies. We define the two-dimensional time series workload model in Section 3. In Section 4, an improved LSTM prediction model is proposed. Section 5 provides a parallel improved LSTM algorithm. In Section 6, we evaluate the proposed prediction model and analyze experimental results. Finally, we conclude the paper and offer some comments on further research in Section 7.

II. RELATED WORK

Prediction technology is a traditional science, widely used in traffic flow [14], power network load [15], tourism management [16], network flow control [17], and many social, economic, natural science fields. The most popularly used time series model is ARIMA [11], [18], and extensions such as seasonal ARIMA models [19] and wavelet ARIMA [20]. Several standard prediction methods are available, including support vector regression (SVR), Bayesian networks, support vector machine (SVM), and others [8].

In recent years, many prediction models have been applied to forecast computing systems workload. Di *et al.* [21] investigated the correlations among Google host workload features, and designed a Bayesian networks prediction model to forecast fluctuation patterns in the host workload. Yang *et al.* [22] used the a linear regression model to predict workload for cloud services. Parameters for this prediction method can adjust dynamically based on fluctuations in service workload. Rahmanian *et al.* [5] combined existing prediction models and proposed an ensemble resource prediction method based on Learning Automata (LA) and clouds theory. Cetinski and Juric [25] combined statistical and learning techniques to improve accuracy in cloud computing workload prediction. In practice, these methods rely on stable historical data to determine model parameters. However, in large-scale computing systems, historical workloads are unstable and nonlinear with regards to their underlying distribution. Therefore, the time series models based on empirical data are not suitable for workload prediction.

To deal with these challenges, many prediction techniques based on machine learning are widely used and have proven to be superior to the mathematical algorithms above [6], [8], [9], [23], [24]. Most of them require a training phase based on large-scale historical data.

Amiri *et al.* [9] in work proposed an online server workload prediction model based on ensemble learning. A feed-forward artificial neural network predictor was provided by Duy *et al.* [23] to forecast the workload in a computational grids host. Qazi and Aizenberg [24] proposed a complex-valued neural network datacenter workload prediction approach. Kumar and Singh [8] combined a self-adaptive differential evolution method and an artificial neural network to predict workload in cloud computing systems. In previous work, we proposed a cloud data center workload prediction model (MLWNN), which is the combination of a wavelet neural network and linear regression [6]. This work yielded an efficient job scheduling strategy based on workload prediction. Neural network-based approaches perform better than classical prediction methods. However, these methods cannot effectively extract features due to the workload long-term spatial and temporal dependency.

Recently, with the Deep learning reporting success by Hinton [26]. RNNs have received increasing attention for their deep structure that connects basic neural units in chronological order. Zhang *et al.* [12] proposed an RNN to predict workload in cloud computing systems. This model uses an orthogonal experimental design to yield the most significant parameters. An LSTM is a specific type of RNN that can effectively learn long-term dependency information. Li and Cao [16] used LSTM neural network to predict tourism flow. Gupta and Dinesh [13] applied a multivariate LSTM model to forecast cloud workload future trends. Their experimental results show that the LSTM prediction model achieves satisfactory performance. However, none of these prediction models fully exploit the two-dimensional characteristics of date and time in the workload of computing systems. Moreover, the handling of real-time constraints for systems resource management is not supported by these methods.

III. DATA PREPROCESSING

The workload in large-scale computing systems such as Supercomputer centers is defined as the ratio of the actual number of computing cores executing various tasks to the aggregate number of such computing resources. The number of aggregate computing resources is usually known in advance. For example, the Shanghai Supercomputer Center has 9000 computing cores available, with an additional 960 reserved for China National Grid [27], [28]. Assessing the number of active cores can be achieved through systems management log files analysis [29]. Using this approach, we can retrieve applications execution information, such as user, application, task, computing cores, start time, finish time, computing nodes. Examples of information are listed in Table 1.

This way, we can ascertain large-scale computing systems active cores within a certain time interval, such as 1 hour, 30 minutes, 10 minutes. Fig. 1 reports Shanghai Supercomputer Center active cores from 7/7/2017 to 7/24/2017, with the time interval set as 1 hour. Here, we retrieved and obtained data for 7200 active cores at the Shanghai Supercomputer Center over 300 days. From this workload dataset, we can conclude that high workloads occur between the times of 8 : 00 and 20 : 00 during the working day, and workload is relatively light during weekends. The workload for a certain working day period is actually likely to be similar to the same period in the previous days or weeks. Following this observation, we believe that the workload in the Shanghai Supercomputer Center has two-dimensional time series characteristics based on day and time.

TABLE 1. Applications execution information.

User	APP	Task	Cores	Start	Finish	Nodes
Rauj	CP2K	BLYP	45	10:23:05	10:56:02	4,8,34,90
Rauj	CP2K	DFT	25	10:43:01	11:07:06	23,45,66
Ahasu	NAMD	MD	89	9:03:11	9:17:19	3,7,8,49,83

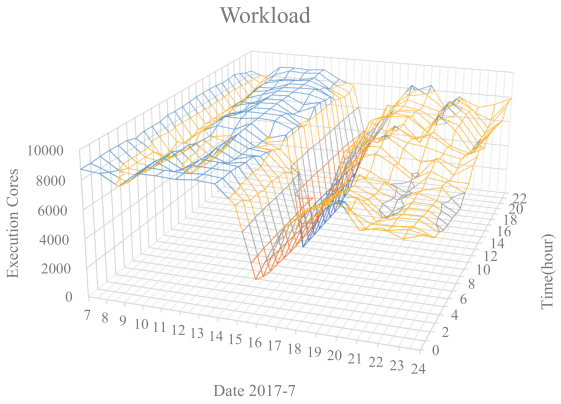

FIGURE 1. Shanghai supercomputer center execution cores.

These two-dimensional time series data are used to train our improved LSTM model and predict future systems workload. Before this, we should smooth out some noisy data related to various uncertainty factors, such as from 7/15/2017 10:00 to 7/15/2017 17:00, when Shanghai Supercomputer Center happened to close for overhaul. Then, we normalize the number of active cores into computing systems workload x, which is defined as following

$$x = \frac{execution\ cores}{total\ cores}. \tag{1}$$

This value is within the range $(0, 1)$ as the number of active cores is usually less than the total number of cores in the system. Due to the two-dimensional characteristic of the workload in the computing system, we label the information as a d day and t time $x^{d,t}$ two-dimensional time series data.

IV. THE IMPROVED LSTM NEURAL NETWORK

The LSTM overcomes the vanishing gradient problem in RNNs and has been proven effective in dealing with long range dependency data, such as computing systems workload [13], [16], [30]. Most importantly, LSTM neural networks introduce three gates: input gate, forget gate, output

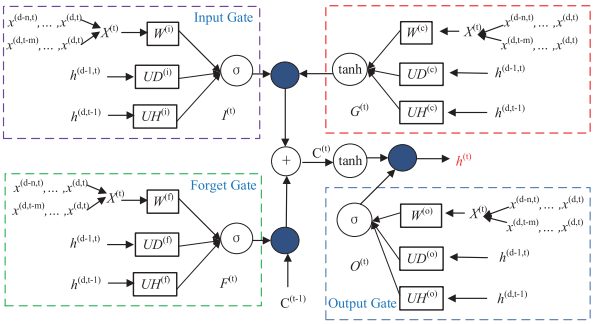

FIGURE 2. The structure of LSTM neural network cells.

gate, and a memory in its cell structure, which is depicted in Fig. 2. Unlike a typical LSTM, each gate in our proposed improved model has a two-dimensional time series d day and t time $x^{d,t}$ workload as input. The previous hidden layer also has two-dimensional hidden information regarding time and day that is transmitted to the next layer.

The main objective of our proposed improved LSTM prediction model is to forecast d day, $t+1$ time workload in large-scale computing systems. Therefore, our proposed improved LSTM uses the previous two-dimensional d day and t time workload $x^{d,t}$ to produce an LSTM neural network input X^t, which is expressed as

$$X^t = [x^{d-n,t}, \cdots, x^{d-1,t}, x^{d,t}] \Lambda [x^{d,t-m}, \cdots, x^{d,t-1}, x^{d,t}]. \tag{2}$$

where X^t are $n+m+2$ input nodes that combined by previous $n+1$ days at time t and $m+1$ computing systems workload time series. This input data is fed into the three gates and the memory of the LSTM neural network cells. Regarding the input gate, forget gate, output gate, and the memory, the implementation includes the following functions:

$$I^t = W^{(i)}X^t + h^{(d-1,t)}UD^{(i)} + h^{(d,t-1)}UH^{(i)}. \tag{3}$$
$$F^t = W^{(f)}X^t + h^{(d-1,t)}UD^{(f)} + h^{(d,t-1)}UH^{(f)}. \tag{4}$$
$$O^t = W^{(o)}X^t + h^{(d-1,t)}UD^{(o)} + h^{(d,t-1)}UH^{(o)}. \tag{5}$$
$$G^t = W^{(c)}X^t + h^{(d-1,t)}UD^{(c)} + h^{(d,t-1)}UH^{(c)}. \tag{6}$$
$$C^t = \sigma(I^t) * tanh(G^t) + \sigma(F^t) * C^{t-1}. \tag{7}$$
$$h^t = \sigma(O^t) * tanh(C^t). \tag{8}$$

where we use the notations i, f, o, and c to denote the input gate, forget gate, output gate, and memory, respectively. The notations I^t, F^t, O^t, and G^t are the outputs of these gates at time t and are regulated by the activation functions. In this improved LSTM, the activation function of input, forget, and output gates is a standard sigmoid function σ, defined as

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{9}$$

The other activation function is *tanh*. The W, UD, UH are the corresponding weights for the input training data, previous hidden layer day-dimensional and time-dimensional value. The notations C and h are the LSTM cell state and hidden layer value, respectively. Finally, the output layer produces a prediction value y^t, which is expressed as

$$y^t = \sigma(W^{(out)}h^t). \tag{10}$$

For t time-step, the standard prediction error $E(t)$ is the actual value $\hat{y}^t$ at time t and $t-1$ with prediction value y^t and y^{t-1}, which is

$$E(t) = \frac{1}{2}[(\hat{y}^t - y^t)^2 + (\hat{y}^{t-1} - y^{t-1})^2]. \tag{11}$$

Then, the improved LSTM leverages this error $E(t)$ to determine the gradient of each weight in the subsequent error back propagation phase. In this study, error back propagation is achieved using the magnitudes of partial derivatives, which allows a determination of the weight after every training iteration. As the partial derivative tied to the hidden layer value h^t is more prevalent in the determination of other weights, we provide it first as

$$\frac{\partial E}{\partial h^t} = W^{(out)}|\hat{y}^t - y^t|\sigma'(W^{(out)}h^t). \tag{12}$$

This h^t partial derivative is core to other weights calculation. Here, the LSTM cell state C^t partial derivative can be defined as

$$\begin{aligned}\frac{\partial E}{\partial C^t} &= \frac{\partial E}{\partial h^t}\frac{\partial h^t}{\partial C^t} \\ &= \frac{\partial E}{\partial h^t} * \sigma(O^t) * (1 - tanh^2(C^t)).\end{aligned} \tag{13}$$

In addition, C^{t-1} receives gradients from h^{t-1} as well as the next cell state C^t. Therefore, back propagation to the hidden layer C^{t-1} partial derivative in an LSTM neural network error is also updated as

$$\begin{aligned}\frac{\partial E}{\partial C^{t-1}} &= \frac{\partial E}{\partial h^{t-1}}\frac{\partial h^{t-1}}{\partial C^{t-1}} + \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial C^{t-1}} \\ &= \frac{\partial E}{\partial h^{t-1}} * \sigma(O^{t-1}) * (1 - tanh^2(C^{t-1})) \\ &\quad + \frac{\partial E}{\partial C^t}\sigma(F^t).\end{aligned} \tag{14}$$

This way, the output layer weight $W^{(out)}$ can be computed as

$$\begin{aligned}\Delta W^{(out)} &= \alpha\frac{\partial E}{\partial W^{(out)}} \\ &= \alpha\left[\frac{\partial E}{\partial y^t}\frac{\partial y^t}{\partial W^{out}} + \frac{\partial E}{\partial y^{t-1}}\frac{\partial y^{t-1}}{\partial W^{out}}\right] \\ &= \alpha\Big[|\hat{y}^t - y^t|\sigma'(W^{(out)}h^t)h^t \\ &\quad + |\hat{y}^{t-1} - y^{t-1}|\sigma'(W^{(out)}h^{t-1})h^{t-1}\Big].\end{aligned} \tag{15}$$

where α is the learning rate, which denotes the step length of each training iteration. With regard to the subsequent calculations for other weights in the error back propagation phase,

the learning rate α is included as well and further explanation on this step is probably unnecessary in this paper. Concerning the three gates and memory in the LSTM neural network cell, their weights W, UD, UH can be then calculated using the following partial derivatives:

$$\begin{aligned}\Delta W^{(i)} &= \frac{\partial E}{\partial W^{(i)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial I^t}\frac{\partial I^t}{\partial W^{(i)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial I^{t-1}}\frac{\partial I^{t-1}}{\partial W^{(i)}} \\ &= \frac{\partial E}{\partial C^t}\sigma'(I^t)tanh(G^t)X^t \\ &\quad + \frac{\partial E}{\partial C^{t-1}}\sigma'(I^{t-1})tanh(G^{t-1})X^{t-1}. \end{aligned} \tag{16}$$

$$\begin{aligned}\Delta UD^{(i)} &= \frac{\partial E}{\partial UD^{(i)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial I^t}\frac{\partial I^t}{\partial UD^{(i)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial I^{t-1}}\frac{\partial I^{t-1}}{\partial UD^{(i)}} \\ &= \frac{\partial E}{\partial C^t}\sigma'(I^t)tanh(G^t)h^{d-1,t} \\ &\quad + \frac{\partial E}{\partial C^{t-1}}\sigma'(I^{t-1})tanh(G^{t-1})h^{d-1,t-1}. \end{aligned} \tag{17}$$

$$\begin{aligned}\Delta UH^{(i)} &= \frac{\partial E}{\partial UH^{(i)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial I^t}\frac{\partial I^t}{\partial UH^{(i)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial I^{t-1}}\frac{\partial I^{t-1}}{\partial UH^{(i)}} \\ &= \frac{\partial E}{\partial C^t}\sigma'(I^t)tanh(G^t)h^{d,t-1} \\ &\quad + \frac{\partial E}{\partial C^{t-1}}\sigma'(I^{t-1})tanh(G^{t-1})h^{d,t-2}. \end{aligned} \tag{18}$$

$$\begin{aligned}\Delta W^{(f)} &= \frac{\partial E}{\partial W^{(f)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial F^t}\frac{\partial F^t}{\partial W^{(f)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial F^{t-1}}\frac{\partial F^{t-1}}{\partial W^{(f)}} \\ &= \frac{\partial E}{\partial C^t}\sigma'(F^t)C^{t-1}X^t \\ &\quad + \frac{\partial E}{\partial C^{t-1}}\sigma'(F^{t-1})C^{t-2}X^{t-1}. \end{aligned} \tag{19}$$

$$\begin{aligned}\Delta UD^{(f)} &= \frac{\partial E}{\partial UD^{(f)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial F^t}\frac{\partial F^t}{\partial UD^{(f)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial F^{t-1}}\frac{\partial F^{t-1}}{\partial UD^{(f)}} \\ &= \frac{\partial E}{\partial C^t}\sigma'(F^t)C^{t-1}h^{d-1,t} \\ &\quad + \frac{\partial E}{\partial C^{t-1}}\sigma'(F^{t-1})C^{t-2}h^{d-1,t-1}. \end{aligned} \tag{20}$$

$$\begin{aligned}\Delta UH^{(f)} &= \frac{\partial E}{\partial UH^{(f)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial F^t}\frac{\partial F^t}{\partial UH^{(f)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial F^{t-1}}\frac{\partial F^{t-1}}{\partial UH^{(f)}} \\ &= \frac{\partial E}{\partial C^t}\sigma'(F^t)C^{t-1}h^{d,t-1} \\ &\quad + \frac{\partial E}{\partial C^{t-1}}\sigma'(F^{t-1})C^{t-2}h^{d,t-2}. \end{aligned} \tag{21}$$

$$\begin{aligned}\Delta W^{(o)} &= \frac{\partial E}{\partial W^{(o)}} \\ &= \frac{\partial E}{\partial h^t}\frac{\partial h^t}{\partial O^t}\frac{\partial O^t}{\partial W^{(o)}} + \frac{\partial E}{\partial h^{t-1}}\frac{\partial h^{t-1}}{\partial O^{t-1}}\frac{\partial O^{t-1}}{\partial W^{(o)}} \\ &= \frac{\partial E}{\partial h^t}\sigma'(O^t)tanh(C^t)X^t \\ &\quad + \frac{\partial E}{\partial h^{t-1}}\sigma'(O^{t-1})tanh(C^{t-1})X^{t-1}. \end{aligned} \tag{22}$$

$$\begin{aligned}\Delta UD^{(o)} &= \frac{\partial E}{\partial UD^{(o)}} \\ &= \frac{\partial E}{\partial h^t}\frac{\partial h^t}{\partial O^t}\frac{\partial O^t}{\partial UD^{(o)}} + \frac{\partial E}{\partial h^{t-1}}\frac{\partial h^{t-1}}{\partial O^{t-1}}\frac{\partial O^{t-1}}{\partial UD^{(o)}} \\ &= \frac{\partial E}{\partial h^t}\sigma'(O^t)tanh(C^t)h^{d-1,t} \\ &\quad + \frac{\partial E}{\partial h^{t-1}}\sigma'(O^{t-1})tanh(C^{t-1})h^{d-1,t-1}. \end{aligned} \tag{23}$$

$$\begin{aligned}\Delta UH^{(o)} &= \frac{\partial E}{\partial UH^{(o)}} \\ &= \frac{\partial E}{\partial h^t}\frac{\partial h^t}{\partial O^t}\frac{\partial O^t}{\partial UH^{(o)}} + \frac{\partial E}{\partial h^{t-1}}\frac{\partial h^{t-1}}{\partial O^{t-1}}\frac{\partial O^{t-1}}{\partial UH^{(o)}} \\ &= \frac{\partial E}{\partial h^t}\sigma'(O^t)tanh(C^t)h^{d,t-1} \\ &\quad + \frac{\partial E}{\partial h^{t-1}}\sigma'(O^{t-1})tanh(C^{t-1})h^{d,t-2}. \end{aligned} \tag{24}$$

$$\begin{aligned}\Delta W^{(c)} &= \frac{\partial E}{\partial W^{(c)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial G^t}\frac{\partial G^t}{\partial W^{(c)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial G^{t-1}}\frac{\partial G^{t-1}}{\partial W^{(c)}} \\ &= \frac{\partial E}{\partial C^t}tanh'(G^t)\sigma(I^t)X^t \\ &\quad + \frac{\partial E}{\partial C^{t-1}}tanh'(G^{t-1})\sigma(I^{t-1})X^{t-1}. \end{aligned} \tag{25}$$

$$\begin{aligned}\Delta UD^{(c)} &= \frac{\partial E}{\partial UD^{(c)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial I^t}\frac{\partial I^t}{\partial UD^{(c)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial I^{t-1}}\frac{\partial I^{t-1}}{\partial UD^{(c)}} \\ &= \frac{\partial E}{\partial C^t}tanh'(G^t)\sigma(I^t)h^{d-1,t} \\ &\quad + \frac{\partial E}{\partial C^{t-1}}tanh'(G^{t-1})\sigma(I^{t-1})h^{d-1,t-1}. \end{aligned} \tag{26}$$

$$\begin{aligned}\Delta UH^{(c)} &= \frac{\partial E}{\partial UH^{(c)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial I^t}\frac{\partial I^t}{\partial UH^{(c)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial I^{t-1}}\frac{\partial I^{t-1}}{\partial UH^{(c)}} \\ &= \frac{\partial E}{\partial C^t}tanh'(G^t)\sigma(I^t)h^{d,t-1} \\ &\quad + \frac{\partial E}{\partial C^{t-1}}tanh'(G^{t-1})\sigma(I^{t-1})h^{d,t-2}. \end{aligned} \tag{27}$$

V. IMPROVED LSTM PARALLELIZATION

The workload prediction model real-time feature is key to resource management in large-scale computing systems. However, traditional LSTM neural networks exhibit inherent computational seriality and require a great deal of processing

time for large-scale data training [31]. Therefore, we propose a parallel improved LSTM algorithm based on shared memory multi-core systems to reduce workload prediction time.

In our proposed improved LSTM neural network model, each training step of the serial algorithm relies on the hidden layer value of the previous day and the directly preceding moment. Furthermore, we test and analyze the improved LSTM serial algorithm data training time, using a processing time of one day as the basic parallel unit. As a result, we only need to break through the improved LSTM day-dimensional dependency to realize the parallelization of the algorithm. In practice, day-dimensional workloads in computing systems exhibit weekly or seasonal periodicity. Therefore, the LSTM cell from a previous hidden layer value $h^{(d-1,t)}$ is changed according to the observation from a week before, $h^{(d-8,t)}$. Fig. 3 indicates this weekly data dependence between 7-7-2017 and 7-14-2017 from the perspective of a day-dimensional workload. Based on this, with each day LSTM training as a parallel unit, we can implement a 7 days processing time parallelization.

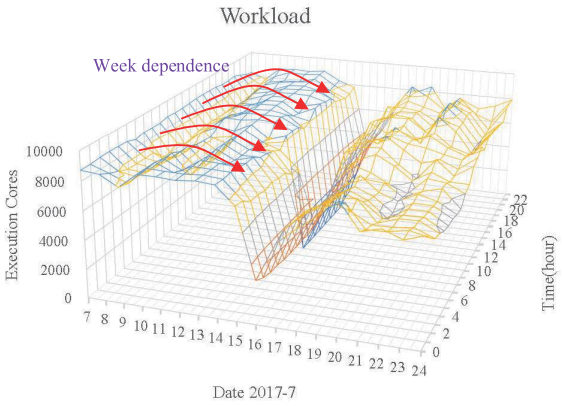

FIGURE 3. The example of week data dependence.

To achieve this parallelization strategy, the Eq. (3), (4), (5), (6) are modified as follows

$$I^t = W^{(i)}X^t + h^{(d-8,t)}UD^{(i)} + h^{(d,t-1)}UH^{(i)}. \quad (28)$$

$$F^t = W^{(f)}X^t + h^{(d-8,t)}UD^{(f)} + h^{(d,t-1)}UH^{(f)}. \quad (29)$$

$$O^t = W^{(o)}X^t + h^{(d-8,t)}UD^{(o)} + h^{(d,t-1)}UH^{(o)}. \quad (30)$$

$$G^t = W^{(c)}X^t + h^{(d-8,t)}UD^{(c)} + h^{(d,t-1)}UH^{(c)}. \quad (31)$$

In the error back propagation phase, the corresponding previous hidden layer day-dimensional weights UD calculation are modified as well as follows

$$\begin{aligned}\Delta UD^{(i)} &= \frac{\partial E}{\partial UD^{(i)}} \\ &= \frac{\partial E}{\partial C^t}\frac{\partial C^t}{\partial I^t}\frac{\partial I^t}{\partial UD^{(i)}} + \frac{\partial E}{\partial C^{t-1}}\frac{\partial C^{t-1}}{\partial I^{t-1}}\frac{\partial I^{t-1}}{\partial UD^{(i)}} \\ &= \frac{\partial E}{\partial C^t}\sigma'(I^t)tanh(G^t)h^{d-8,t} \\ &\quad + \frac{\partial E}{\partial C^{t-1}}\sigma'(I^{t-1})tanh(G^{t-1})h^{d-8,t-1}. \quad (32)\end{aligned}$$

Algorithm 1 The Pseudocode of Improved LSTM Weights W, UD, UH Parallelization Algorithm

```
Initialize parallel sections;
Set private list weights W, UD, UH;
while the training data are not empty do
    for each parallel unit parallel do
        Copy private list weights W, UD, UH from
        Master;
        LSTM training and compute weights iteratively;
        Use Eq. (33) to compute parallel unit error;
        Barrier;
        Copy the weights W, UD, UH of parallel unit
        with minimum pe to Master;
    end
end
```

The other parallelization constraints in this improved LSTM are the training weights W, UD, UH, which are shared by all training procedures. Here, we propose an improved LSTM weights parallelization algorithm, which is formalized in **Algorithm 1**. To achieve this goal, we define a parallel unit error pe, which is expressed as

$$pe = \sum E_d(t). \quad (33)$$

where pe is the aggregate improved training error of parallel units in the LSTM neural network. The **Algorithm 1** first copies master LSTM weights into parallel unit private variables. Each parallel unit then completes LSTM training and iteratively ascertains weights. Finally, this algorithm uses Eq.(33) to compute the parallel unit error pe, and find parallel unit weights with a minimum pe as master weights.

VI. PERFORMANCE EVALUATION

To assess the performance of our proposed improved LSTM neural network prediction model, we conducted experiments on a SuperMicro 8046B-TRF with an eight-core Xeon X7550. The training and evaluation workload data came from the Shanghai Supercomputer Center actual active cores over 300 days [27]. This large-scale computing system has 9000 cores [27]. We used the first 290 days sample for training the improved LSTM neural network prediction model and its parallel algorithm, and time series data for the next 10 days was directed toward evaluation.

This work compares the improved LSTM neural network prediction model with traditional LSTM [13] to evaluate its effectiveness in large-scale computing systems. Mean square error (MSE) is used as a loss function to evaluate these models [12], [13], which is given as:

$$MSE = \frac{\sum_{t=1}^{T}(\hat{y}^t - y^t)^2}{T}. \quad (34)$$

where T is the number of time series in evaluation phase.

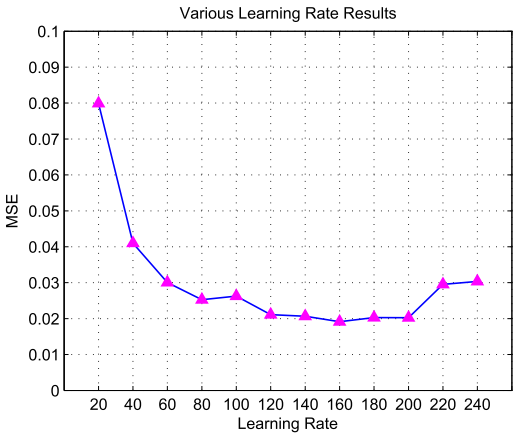

FIGURE 4. Various learning rate experimental results.

A. VARIOUS LEARNING RATE

In the first set of experiments, we evaluated the performance for various learning rates α with our proposed improved LSTM prediction model. Fig. 4 indicates the experimental results for computing systems when varying the learning rate α from 20 to 240, using steps of 20. We observe from Fig. 4 that the mean square error (*MSE*) decreases as the learning rate α increases from 20 to 160. However, as α continues to increase, the performance of our proposed prediction model deteriorates (*MSE* increases). Therefore, we think that the optimal learning rate α is 160. In the subsequent experiments, we used the learning rate of $\alpha = 160$ to compare performance.

B. THE IMPROVED LSTM PERFORMANCE RESULTS

The second set of experiments attempted to compare the improved LSTM prediction model with traditional LSTM, with the results shown in Fig. 5. The performance evaluation data was the actual workload within the Shanghai Supercomputer Center from 2/11/2018 to 2/20/2018.

From Fig. 5, we can conclude that our proposed improved LSTM neural network prediction model performs better than a traditional LSTM. In fact, the average error for the improved LSTM compared with real-world workload is 6.26%, while the corresponding metric for a traditional LSTM is 19.25%. Therefore, our proposed improved LSTM outperforms traditional LSTM by 67.5%. The performance metric mean square error (*MSE*) also supports this conclusion. Indeed, the *MSE* of the improved LSTM prediction model is 0.019 and while the corresponding value for a traditional LSTM is 0.042. Therefore, our proposed improved LSTM outperforms a traditional LSTM by 54.8% in terms of mean square error. This improvement is mainly attributable to the adoption in the improved LSTM of a two-dimensional time series LSTM neural network cell structure, which is very suitable to the workload characteristics in large-scale computing systems.

TABLE 2. Parallel improved LSTM experimental results.

Cores	*Speedup*	Cores	*Speedup*
2	1.45	4	2.98
6	4.05	8	4.76

C. THE PARALLEL MODEL PERFORMANCE COMPARISON

This Section attempts a comparative analysis of our proposed parallel improved LSTM prediction model. We conducted these experiments on a multi-core processor and varied cores from 2 to 8, in steps of 2. Table 2 lists the *Speedup* in the parallel improved LSTM. The *Speedup* is computed by dividing the serial algorithm execution time by the parallel algorithm execution time, which is a standard performance evaluation

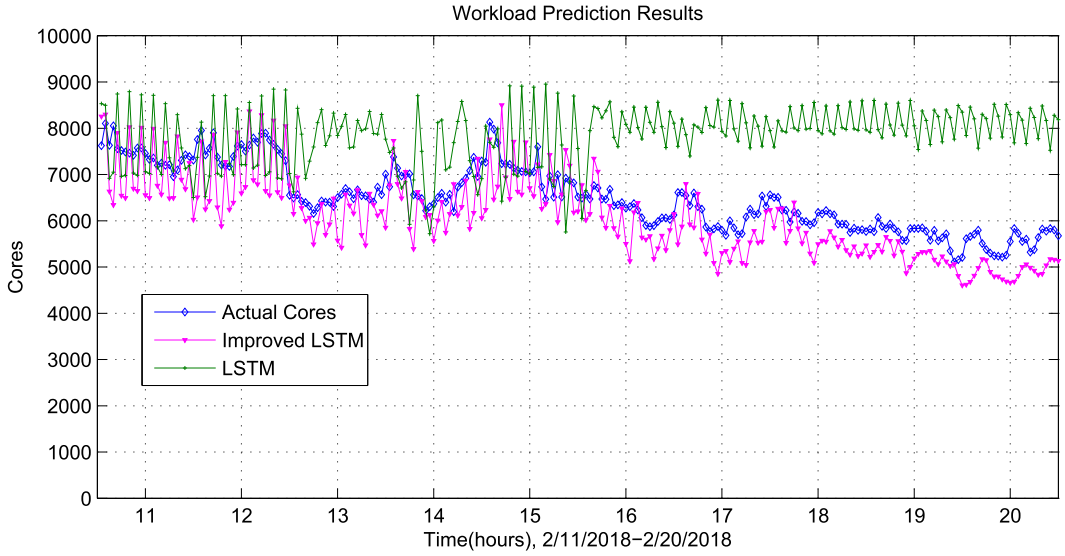

FIGURE 5. The improved LSTM performance comparisons.

metric for parallel algorithms. From Table 2, we observe that the proposed parallel improved LSTM prediction model can yield an improvement in *Speedup* of 4.76 times over a serial algorithm as the number of cores reaches 8.

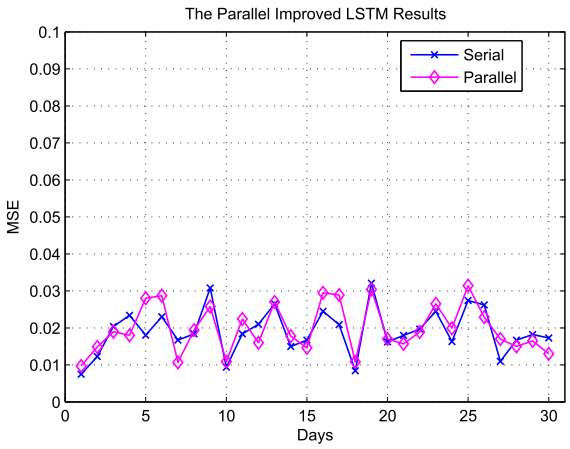

FIGURE 6. The experimental results of parallel and serial improved LSTM.

The other performance comparison focuses on the accuracy of these two algorithms. The experimental results are shown in Fig. 6, with as plot of the daily mean square error for 30 days of evaluation. We observe from Fig. 6 that the *MSE* for parallel and serial improved LSTM prediction models are also similar. In fact, the total *MSE* for serial algorithms is 0.574802, while it reaches 0.5963 for the parallel algorithm. It appears that the parallel improved LSTM prediction model is inferior to the serial algorithm by a narrow margin of 3.6%. We believe achieving a higher workload prediction speed while only suffering a moderate loss in prediction accuracy is very valuable in the context of resource management for large-scale computing systems.

VII. CONCLUSIONS AND FUTURE WORK

Resource management and workload prediction have always been challenges in large-scale computing systems. In this study, we attempted to improve the effectiveness of resource management by enhancing the precision of workload prediction. To achieve this goal, we first analyzed the interdependence in historical workload information for large-scale computing systems and built a day and time two-dimensional time series workload model. We then proposed an improved LSTM neural network prediction model using an LSTM cell structure aimed at a two-dimensional time series and its corresponding mathematical description. Finally, a parallel improved LSTM algorithm was proposed using a hidden layer based on weekly dependence and weights W, UD, UH for the parallelization algorithm.

The performance of the improved LSTM workload prediction model was evaluated with the actual workload of the Shanghai Supercomputer Center over 300 days. The experimental results clearly confirmed the superior performance of the proposed improved LSTM over a traditional LSTM. In particular, the parallel improved LSTM can yield better workload prediction accuracy and faster prediction speed on large-scale computing systems.

This work is one of the first attempts to use neural networks to forecast computing systems workload. It will be interesting in future studies to explore managing system resources, energy, and user jobs based on workload prediction.

ACKNOWLEDGMENTS

The authors would like to thank the Shanghai Supercomputer Center, Genguo Li, Dazhi Kou that provide the systems workload log files.

REFERENCES

[1] D. Roten *et al.*, "High-frequency nonlinear earthquake simulations on petascale heterogeneous supercomputers," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2016, pp. 957–968.
[2] A. Gutierrez-Milla, M. Mantsinen, M. Avila, G. Houzeaux, C. Riera-Auge, and X. Sáez, "New high performance computing software for multiphysics simulations of fusion reactors," *Fusion Eng. Des.*, vol. 136, pp. 639–644, Nov. 2018.
[3] *The Supercomputers Tianhe-2A Maximum Operating Power Consumption*. Accessed: Jan. 20, 2019. [Online]. Available: https://www.top500.org/lists/2018/11/
[4] D.-J. Lim, T. R. Anderson, and T. Shott, "Technological forecasting of supercomputer development: The march to exascale computing," *Omega*, vol. 51, pp. 128–135, Mar. 2015.
[5] A. A. Rahmanian, M. Ghobaei-Arani, and S. Tofighya, "A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment," *Future Gener. Comput. Syst.*, vol. 79, pp. 54–71, Feb. 2018.
[6] X. Tang, X. Liao, J. Zheng, and X. Yang, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Comput.*, vol. 21, no. 3, pp. 1581–1593, Sep. 2018.
[7] B. Liu, Y. Lin, and Y. Chen, "Quantitative workload analysis and prediction using Google cluster traces," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 935–940.
[8] 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. 41–45, Apr. 2018.
[9] M. Amiri, L. Mohammad-Khanli, and R. Mirandola, "An online learning model based on episode mining for workload prediction in cloud," *Future Gener. Comput. Syst.*, vol. 87, pp. 83–101, Oct. 2018.
[10] N. Singh and S. Rao, "Online ensemble learning approach for server workload prediction in large datacenters," in *Proc. 11th Int. Conf. Mach. Learn. Appl.*, 2012, pp. 68–71.
[11] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining Kohonen maps with ARIMA time series models to forecast traffic flow," *Transp. Res. C, Emerg. Technol.*, vol. 4, no. 5, pp. 307–318, Oct. 1996.
[12] W. Zhang, B. Li, D. Zhao, F. Gong, and Q. Lu, "Workload prediction for cloud cluster using a recurrent neural network," in *Proc. Int. Conf. Identificat., Inf. Knowl. Internet Things (IIKI)*, 2016, pp. 104–109.
[13] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Dec. 2017, pp. 1–6.
[14] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.
[15] C. Xia, M. Zhang, and J. Cao, "A hybrid application of soft computing methods with wavelet SVM and neural network to electric power load forecasting," *J. Elect. Syst. Inf. Technol.*, vol. 5, no. 3, pp. 681–696, Dec. 2018.
[16] Y. Li and H. Cao, "Prediction for tourism flow based on LSTM neural network," *Procedia Comput. Sci.*, vol. 129, pp. 277–283, Dec. 2018.
[17] R. Madan and P. SarathiMangipudi, "Predicting computer network traffic: A time series forecasting approach using DWT, ARIMA and RNN," in *Proc. 11th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2018, pp. 1–5.
[18] X. Tang and X. Liao, "Application-aware deadline constraint job scheduling mechanism on large-scale computational grid," *PLoS ONE*, vol. 13, no. 11, Nov. 2018, Art. no. e0207596.

[19] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results," *J. Transp. Eng.*, vol. 129, no. 6, pp. 664–672, Nov. 2003.
[20] A. R. Syed, S. M. Burney, and B. Sami, "Forecasting network traffic load using wavelet filters and seasonal autoregressive moving average model," *Int. J. Comput. Elect. Eng.*, vol. 2, no. 6, pp. 1793–8163, Dec. 2010.
[21] S. Di, D. Kondo, and W. Cirne, "Google hostload prediction based on Bayesian model with optimized feature combination," *J. Parallel Distrib. Comput.*, vol. 74, no. 1, pp. 1820–1832, Jan. 2014.
[22] J. Yang *et al.*, "A cost-aware auto-scaling approach using the workload prediction in service clouds," *Inf. Syst. Front.*, vol. 16, no. 1, pp. 7–18, Mar. 2014.
[23] T. V. Duy, Y. Sato, and Y. Inoguchi, "Improving accuracy of host load predictions on computational grids by artificial neural networks," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, May 2009, pp. 1–8.
[24] K. Qazi and I. Aizenberg, "Towards quantum computing algorithms for datacenter workload predictions," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 900–903.
[25] K. Cetinski and M. B. Juric, "AME-WPC: Advanced model for efficient workload prediction in the cloud," *J. Netw. Comput. Appl.*, vol. 55, pp. 191–201, Sep. 2015.
[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[27] *Shanghai Supercomputer Center Computing Cores*. Accessed: Jan. 12, 2019. [Online]. Available: http://www.ssc.net.cn/
[28] *China National Grid Computing Cores*. Accessed: Jan. 17, 2019. [Online]. Available: http://www.cngrid.org/
[29] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "A lightweight algorithm for message type extraction in system application logs," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 11, pp. 1921–1936, Nov. 2012.
[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[31] K. Hwang and W. Sung, "Single stream parallelization of generalized LSTM-like RNNs on a GPU," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1047–1051.

XIAOYONG TANG received the M.S. and Ph.D. degrees from Hunan University, China, in 2007 and 2013, respectively. He has published more than 30 technique papers in international journals and conferences. His research interests include neural networks, distributed computing systems scheduling, parallel computing, and parallel algorithms. He is a reviewer of TC, TPDS, JPDC, FGCS, JS, and so on.