Predicting Inter-Data-Center Network Traffic Using Elephant Flow and Sublink Information

Yi Li, *Student Member, IEEE*, Hong Liu, Wenjun Yang, Dianming Hu, Xiaojing Wang, and Wei Xu

***Abstract*—With the ever increasing number of large scale Internet applications, inter-data-center (inter-DC) data transfers are becoming more and more common. Traditional inter-DC transfers suffer from both low utilization and congestion, and traffic prediction is an important method to optimize these transfers. Inter-DC traffic is harder to predict than many other types of network traffic because it is dominated by a few large applications. We propose a model that significantly reduces the prediction errors. In our model, we combine wavelet transform with artificial neural network to improve prediction accuracy. Specifically, we explicitly add information of sublink traffic and elephant flows, the least predictable yet dominating traffic in inter-DC network, into our prediction model. To reduce the amount of monitoring overhead for the elephant flow information, we add interpolation to fill in the unknown values in the elephant flows. We demonstrate that we can reduce prediction errors over existing methods by 5%∼30%. Our prediction is in production as part of the traffic scheduling system at Baidu, one of the largest Internet companies in China, helping to reduce the peak network bandwidth.**

***Index Terms*—Datacenter network, network management, network traffic prediction, elephant flow, sublink.**

I. Introduction

LARGE scale and geographically distributed applications are on the rise. These applications, such as Web search, video streaming and file sharing are commonly distributed to several data centers. Partitioning applications into multiple data centers can help reducing cost and improve service reliability.

All these applications can lead to heavy network traffic among the data centers. We call this type of traffic *inter-data-center* (inter-DC) traffic to differentiate it from the traffic from end-users accessing these applications from the Internet (which we call *Internet traffic*). Many large service providers use dedicated fibers either owned or leased to handle inter-DC traffic [4]. Given the cost of inter-DC bandwidth, it is essential to keep the inter-DC links highly utilized.

Many Internet service providers (ISPs) charge for bandwidth by the peak bandwidth that a customer uses. Pure traffic shaping might be useful to reduce the peak bandwidth but it may hurt the performance of some critical applications (esp. when the priority is not configured correctly). Thus scheduling traffic over multiple links available with traffic engineering methods to reduce peak bandwidth of each link is important to reduce costs. Existing work such as Google's B4 and Microsoft's SWAN uses software defined network (SDN) to accurately monitor and schedule the inter-DC data transfers. However, most conventional data centers do not have the infrastructure to support flow-level monitoring and scheduling, and thus rely on an accurate prediction of the future traffic to perform short-term / long-term traffic scheduling.

With the high utilization of inter-DC links, spikes and fluctuations in the traffic can cause congestions, which are especially harmful to interactive applications. Accurate network traffic prediction is an important component for tasks like network resource provisioning, scheduling and traffic engineering [1]. For example, if we can predict short-term traffic patterns (1 minute or less), we can move latency sensitive flows out of a network link, if we predict that the link will be congested soon. This is especially helpful as we always have multiple redundant paths between data centers. If we can predict even longer-term traffic, say 30 minutes, we can help the job scheduler to decide if it should delay some batch jobs to further reduce network bandwidth cost. Thus traffic prediction has been a hot research topic. However, to our knowledge, there is no work taking the special inter-DC traffic patterns into account.

In this paper, we present our new model for predicting the network traffic on inter-DC links at Baidu, one of largest Internet companies in China. The links serve as Baidu's inter-DC backbone, connecting multiple data centers with tens of thousands of servers. These data centers host hundreds of large scale applications, both interactive and batch. Using our prediction method, we can reduce the prediction errors by 10%∼30% and Baidu is able to reduce the peak bandwidth for about 9% on average.

While researchers have proposed many network prediction models under different network environments, these models

Manuscript received March 15, 2016; revised June 30, 2016; accepted July 2, 2016. Date of publication July 7, 2016; date of current version December 8, 2016. This research is supported in part by the National Natural Science Foundation of China Grants 61361136003, 61379088, China 1000 Talent Plan Grants, Tsinghua Initiative Research Program Grants 20151080475, and a Google Faculty Research Award. The associate editor coordinating the review of this paper and approving it for publication was L. Granville. *(Corresponding author: Wei Xu.)*

Y. Li and W. Xu are with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China (e-mail: li-yi13@mails.tsinghua.edu.cn; weixu@mail.tsinghua.edu.cn).

H. Liu and X. Wang are with Baidu Inc., Beijing 100083, China (e-mail: liuhong03@baidu.com; wangxiaojing@baidu.com).

W. Yang was with Baidu Inc., Beijing 100083, China. He is now with Didi Inc., Beijing 100085, China (e-mail: yangwenjunreo@didichuxing.com).

D. Hu was with Baidu Inc., Beijing 100083, China. He is now with NovuMind Inc., Beijing 100193, China (e-mail: dianming.hu@gmail.com).

Digital Object Identifier 10.1109/TNSM.2016.2588500

do not work well for inter-DC traffic prediction. There are several reasons why it is hard to predict inter-DC traffic.

First, inter-DC traffic neither represents linear processes nor has stable statistical properties, thus widely used linear models for time-series prediction, such as Autoregressive models (AR) [17], Autoregressive moving average models (ARMA) [18] and Autoregressive Integrated Moving Average models (ARIMA) [20] do not work well. As we will show in Section IV, much of the inter-DC traffic exhibits a highly non-regular and non-linear pattern, mainly because of the existence of many network-resource hungry applications. Linear methods like ARIMA, though proven good for Internet traffic, not enough for inter-DC traffic. Our evaluation confirms the fact. Thus, we need some models to capture the non-linearity in the data.

Second, inter-DC traffic exhibits different patterns compared to Internet backbone traffic. Studies have shown that data center traffic is bursty and unpredictable at such long time-scales (especially at 100 seconds or longer timescales) [2]. In fact, with the data collected at Baidu, we can predict the Internet traffic 10 minutes ahead with only about 2% error from the real value. However, the inter-DC traffic prediction error is as high as 8% to 9%.

Third, the recurring patterns in inter-DC traffic are not obvious because this traffic is often generated by a small number of large applications. For example, in our case, the top 5 applications account for about 80% of the inter-DC traffic. The elephant flows generated by these applications usually occupy large portion of traffic [24] and impact more on the total traffic than mice flows. Usually, the number of elephant flows is far smaller than the number of mice flows, which is referred to as "the elephants and mice phenomenon" [8]. A network flow is called an *elephant flow* if it occupies a large proportion of network traffic. Specifically, in this paper, we take the largest few flows that contribute to at least 80% of the total traffic as elephant flows. In our application, we only observe five applications producing such flows.

Fourth, usually a data center is connected to multiple other data centers and thus the incoming/outgoing traffic is contributed by many data centers. If a link connects two data centers and its traffic contributes to the total traffic of one data center, we call it a *sublink*, as Figure 1 shows. A sublink not only carries the traffic between two data centers, but also acts as a bypass for other data centers, thus different sublinks may reveal different traffic patterns and they have different impacts on the total incoming/outgoint traffic, which makes the total traffic more unpredicable.

There are five key ideas in our prediction method.

First, we apply wavelet transform [22] to decompose the raw time domain traffic to capture both the time and frequency features. We apply Daubechie's 4 (Db4) wavelets with ten levels of decomposition [23] and show that it works well in reducing prediction errors.

Second, we put incoming and outgoing traffic together for training. Thus we can predict incoming and outgoing traffic using a single model, greatly saving the model training time.

Third, we recognize the contribution of elephant flows to the inter-DC traffic. We explicitly add information about elephant flows as separate feature dimensions in the prediction. A practical difficulty is that it is quite expensive to capture all elephant flow information frequently enough to help with the short term prediction. We use different interpolation methods to fill in the missing values of elephant flow traffic, which allow us to incorporate elephant flow information without introducing much data collection overhead.

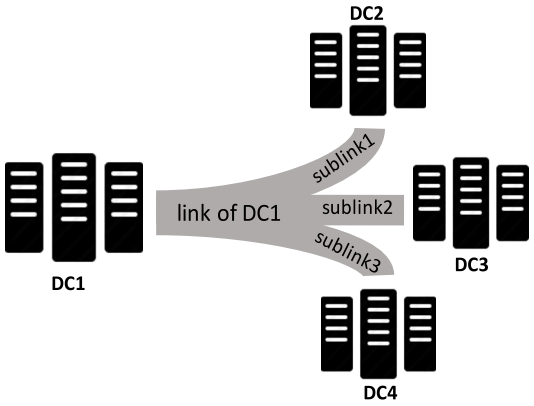

Fig. 1. The view of sublinks. A sublink connect two data centers and the network traffic of the link of DC1 in the figure is contributed by traffic of many sublinks. We can use sublink traffic information to help predict the total incoming/outgoing traffic of the link of DC1.

Fourth, we also add sublink traffic information as dimensions for training to separate the impacts of different data centers so that our model can capture more features of the total traffic.

Last but not least, as the patterns are highly non-linear, we use artificial neural network (ANN) to build the prediction model. ANN not only handles non-linearity well, but also allows us to combine different features into the same model.

Note that all the features from wavelet transform, elephant flows and sublink traffic can be regarded as decompositions. The wavelet transformation is an internal decomposition as we are decomposing the traffic time series using the series itself, while separating out the elephant traffic and sublink traffic is an example of external decomposition using additional information. Combining the internal and external decomposition is the key for our prediction accuracy improvements.

We make the following three contributions:

1) We propose a network traffic prediction model for inter-DC traffic, a traffic type that is hard to predict using previous models, by including the elephant flow and sublink information into our model explicitly. We show that by combining wavelet transform and artificial neural networks, we can reduce prediction errors significantly.

2) We introduce effective interpolation methods to reduce the amount of expensive flow-level observations for the elephant flows.

3) We evaluate our model on a real world, massive scale inter-DC network with tens of thousands of servers and reduce prediction errors by 10%∼30% over existing work.

The rest of this paper is organized as follows. Section II presents the researches on network traffic prediction in recent years. Section III describes our model. Section IV shows the

experiment results, including comparisons between different strategies. We conclude in Section V.

II. Related Work

Many studies have been done on network traffic prediction with traditional linear models. Hu *et al*. [26] used Seasonal Trend Decomposition using Loess (STL) [21] to decompose original series into three components: season component, trend component and irregular component and then used X11-ARIMA for network traffic prediction. Yoo and Sim [10] developed a model to support prediction on high-bandwidth network. FARIMA, known as autoregressive fractionally integrated moving average, which captures the characters of long-memory time series, is also widely used in traffic prediction [28]. Zhou *et al*. [27] combined ARIMA and GARCH, which is a non-linear model, to create a conditional mean and conditional variance model called ARIMA/GARCH, and compared the differences of the performance between ARIMA/GARCH and FARIMA. Periyanayagi and Sumathy [29] proposed a time series model called S-ARMA, using Swarm intelligence and ARMA, for the network traffic prediction in wireless sensor networks. Wavelet transform have been used to preprocess series before the prediction with linear models [7], [32]. However, as inter-DC traffic is bursty and unpredictable at long-time scales, linear models are not suitable for inter-DC traffic prediction, especially for long-time-ahead prediction.

Learning methods are useful in network traffic prediction. Researchers have applied Support Vector Machine (SVM) based classification and regression for time series prediction. For example, Feng *et al*. [33] applied SVM for one-step-ahead prediction on WLAN and compared the performance for various prediction methods. Qian *et al*. [34] used Empirical Mode Decomposition (EMD) to reduce the noise in the data before applying SVM for prediction.

Another important and useful learning model for time-series prediction is artificial neural networks (ANNs) [15]. ANNs have the capability to do non-linear modeling and approximate any continuous function to any desired accuracy theoretically [19], thus ANNs can be used to predict complex time series. Some variants of ANNs have been proposed. For example, algorithms such as PSO [6] can be used to optimize the training process. We can also embed new tools such as wavelet transformation into a neural network, like [12] did. Zhang [14] proposed a hybrid approach to time series forecast using both linear ARIMA model and the nonlinear ANN to predict complex series data with both linear and nonlinear correlation structures. Wavelet Neural Network (WNN) employs nonlinear wavelet basis functions to solve nonlinear fitting problems and have been used for traffic prediction [12]. Xiao *et al*. [36] studied fuzzy-neural network prediction models with wavelet decomposition. Alarcon-Aquino and Barria [13] combined maximal overlap discrete wavelet transform (MODWT) with ANNs and proposed a multi-resolution finite-impulse-response (FIR) neural-network-based learning algorithm, which would be suitable for capturing low- and high-frequency information as well as the dynamics of time-varying signals.

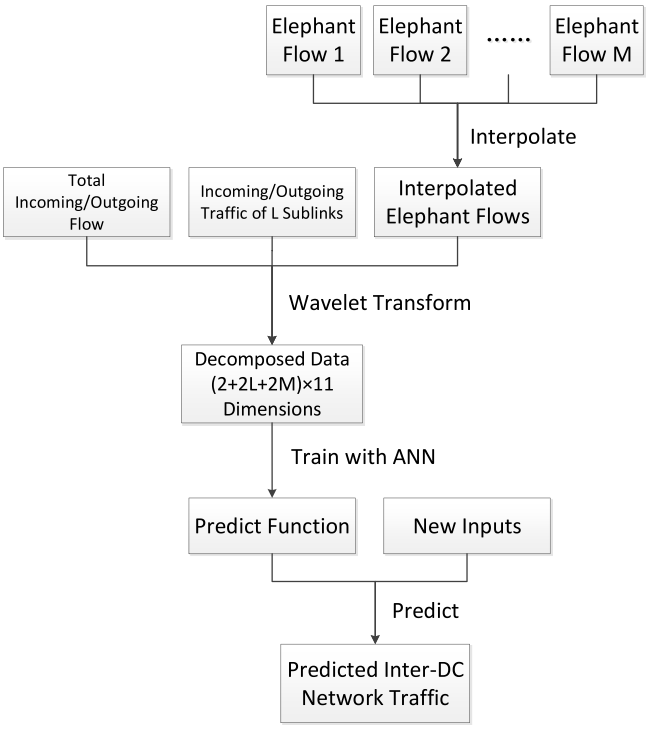

Fig. 2. The process flow of our model. After training, when we get a new data item, which contains the total incoming and outgoing traffic, the sublink traffic and the sampled or interpolated traffic data of elephant flows, we perform decomposition to it. Then we take the features of previous k steps, including this step, as the input of the predict function, and get the predicted future total traffic.

On our inter-DC traffic dataset, we experimented different prediction models, and did not find significant improvements on prediction accuracy. It is not coincidental: the inter-DC traffic is consisted of traffic of different sublinks and dominated by a combination of elephant flows, which demonstrates less patterns. In this work, instead of keep improving the prediction models, we focus on designing better features to capture information of the elephant flow and link traffic.

III. Model Overview

In our model, we collect the total incoming/outgoing traffic data, traffic data of elephant flows and traffic data of different sublinks. As the traffic data of elephant flows is sampled less frequently than the total traffic and sublink traffic, we use interpolation methods to construct the missing values so that we can align total traffic data samples with that of the elephant flows. Then we decompose the collected data with wavelet transform to reveal additional frequency information for training. After decomposition, we normalize the data and train it with ANN to get a prediction function. With the prediction function and new inputs, we can predict the total incoming/outgoing traffic data in near future. Figure 2 shows the process flow of our model.

A. Data Collection

We collect three types of data from each inter-DC link: the total traffic for both incoming and outgoing directions,

a sample of elephant flows and the traffic of sublinks for both directions. Given a time series $(t_1, t_2, \ldots, t_n)$, we denote the total incoming/outgoing traffic at time t_i as in^i and out^i. To reduce useless information and improve the efficiency of computation, we only use information from the top M applications which account for a great proportion of total traffic (we use 80% in this paper). We use a $2M$-dimensional vector to represent the raw elephant flow information at each sample time: $(ein_1, eout_1, ein_2, eout_2, \ldots, ein_M, eout_M)$, where ein_k and $eout_k$(where $k = 1, 2, \ldots, M$) denotes the number of incoming/outgoing traffic of the k-th largest application. To represent the traffic values of sublinks, we use a $2L$-dimensional vector: $(sin_1, sout_1, sin_2, sout_2, \ldots, sin_L, sout_L)$, where L is the number of sublinks that connects other data centers with this data center.

As the traffic data of elephant flows is sampled less frequently, we use interpolation methods to construct the missing values to roughly align the data sample of the total traffic and the elephant flow samples. Thus for each timestamp t_i, we get a $(2 + 2M + 2L)$-dimensional vector as our raw data: $(in^i, out^i, ein^i_1, eout^i_1, ein^i_2, eout^i_2, \ldots, ein^i_M, eout^i_M, sin_1, sout_1, sin_2, sout_2, \ldots, sin_L, sout_L)$.

The goal of the prediction is that given all the history, we want to predict the traffic at different time points in the near future. Formally, we want to predict the next k-step total traffic tuple (in^{i+k}, out^{i+k}), where $k = 1, 2, \ldots$.

Note that we have an alternative approach to model the incoming and outgoing traffic separately, using two $(1 + M + L)$-dimensional vectors for each. Intuitively, the incoming and outgoing traffic of a data center are highly correlated. Using a combined model can help us save model training cost by about 40% while not affecting the prediction accuracy much. We compare these two models in Section IV.

B. Interpolation

The elephant flow data are sampled less frequently to reduce the resource cost. We fill in the missing values using interpolation, a common method in numerical analysis. There are many interpolation methods. In this paper, we compare the following four methods.

One of the simplest methods is *zero interpolation*, which fills zeros for all unknown points. Surprisingly, even with this simple method, we can still significantly reduce the prediction errors compared to methods without using elephant flow information.

We call the second method *scale interpolation*. As the elephant flows occupy large part of the total traffic, we construct the missing values by filling in a number that is proportional to the total traffic. Given the total incoming traffic in^i and in^{i+s} at time t_i and t_{i+s} respectively, assume that the traffic data of elephant flows is sampled at that two time points but not sampled at $t_{i+1}, t_{i+2}, \ldots, t_{i+s-1}$. Then the unsampled incoming traffic $ein^{i+s'}_k (0 < s' < s)$ of application k, the interpolation value, namely the incoming traffic of the elephant flow, is

$$ein^{i+s'}_k = ein^i_k \times \frac{in^{i+s'}}{in^i}.$$

Intuitively, this method may reduce the effectiveness of adding elephant flow information, as we "pollute" the elephant flow data with numbers that is highly correlated with the total traffic, and our experiments confirm the intuition.

The third method is *linear interpolation*. Using the same notations as above, the interpolation value is

$$ein^{i+s'}_k = ein^i_k + (t_{i+s'} - t_i) \times \frac{ein^{i+s}_k - ein^i_k}{t_{i+s} - t_i}$$

which means $(t_{i+s'}, ein^{i+s'}_k)$ is a point in a line segment linking (t_i, ein^i_k) and (t_{i+1}, ein^{i+s}_k).

The last interpolation method we use is *spline interpolation*. With spline interpolation, we can get a smooth curve linking points. To make the interpolation error small and make the computation simple, we decide to use third order polynomials as interpolation functions (also known as *cubic spline interpolation*) [25]. It is more complex than zero interpolation, scale interpolation and linear interpolation. With spline interpolation, we can get a smooth curve linking points. Given different kinds of spline functions, we can get different kinds of spline interpolations. Then the interpolation function $S(t)$, which is used to construct the missing values of the incoming traffic of k-th elephant flow, is a piecewise function:

$$S(t) = \begin{cases} S_0(t) & t \in [t_0, t_{s_1}] \\ S_1(t) & t \in [t_{s_1}, t_{s_2}] \\ \quad\vdots & \\ S_m(t) & t \in [t_{s_m}, t_n] \end{cases}$$

where t_0, t_{s_1}, $t_{s_2}, \ldots, t_{s_m}$ and t_n are the time points at which the elephant flows are sampled.

To ensure the curve is smooth, for each adjacent point pair (t_{s_i}, ein_{ks_i}) and $(t_{s_{i+1}}, ein_{ks_{i+1}})$, where $k = 1, 2, \ldots, n$, and the corresponding interpolation function S_i, we have

$$\begin{cases} S_i(t_{s_i}) = ein_{ks_i} \\ S_i(t_{s_{i+1}}) = ein_{ks_{i+1}} \\ S'_i(t_{s_i}) = S'_{i-1}(t_{s_i}) \quad (i \geq 1) \\ S''_i(t_{s_i}) = S''_{i-1}(t_{s_i}) \quad (i \geq 1) \end{cases}$$

For each i, S_i is a third order polynomial of t, which means S_i can be written in the form of $a_i t^3 + b_i t^2 + c_i t + d_i$, where a_i, b_i, c_i and d_i are constants. Thus, there are total $4n$ unknowns and $4n - 2$ equations. With some initial conditions, such as $S''_1(0) = S''_m(t_n) = 0$, we can solve the equation sets and get the smooth curve passing through given points. With $S(t)$, we have $ein^*_{ki} = S(t_i)$.

Using interpolations allows us to use the elephant flow information while keeping the monitoring cost low. We evaluated all four kinds of interpolations and show the results in Section IV.

C. Decomposition

As we use learning algorithms to predict the traffic, we need "features" (in machine learning terminology) to capture the predictable information at each time point. We use decomposition to provide better features.

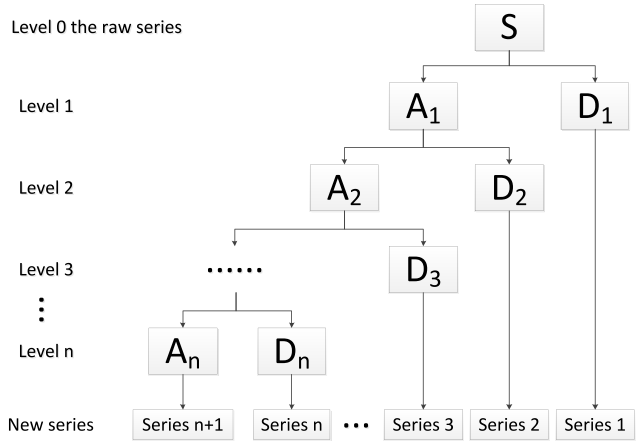

Fig. 3. Wavelet Decomposition. A_i is the low-frequency part at level i, which represents the trend of the series, while D_i is the high-frequency part at level i, which represents the details of the series. After the preprocess, we can get $n+1$ new time series: $D_1, D_2, \ldots, D_n, A_n$.

We decompose the raw data into new series using wavelet transform, which extract deeper information from the raw data. Wavelet transform is a powerful technique to analyze time series. Comparing to Fourier transform, wavelet transform has advantages in processing time-domain series data as it can reserve both time and frequency information while Fourier transform can only reserve frequency information. Wavelet transform uses wavelet functions to decompose time series. A wavelet is a function Ψ that is used to decompose the time series to a low-frequency part and a high-frequency part:

$$X(a,b) = \frac{1}{\sqrt{b}} \int_{-\infty}^{+\infty} x(t) \Psi\left(\frac{t-a}{b}\right) dt$$

where a is the scaling parameter and b is the translation parameter. In practice, we use discrete wavelet (DWT) instead of continuous wavelet transform (CWT) as CWT computation is much more expensive than DWT. We recursively decompose the low-frequency part, adding one new series per recursive run. This process continues to produce more new series until some conditions, such as enough number of levels, are satisfied. For example, we choose to 10-level decomposition in our paper, following the choice in [23]. Fig. 3 shows this.

Assuming we use wavelet transform with w levels of decomposition, we decompose each series in the raw data into $w+1$ new series. We directly feed all the $w+1$ dimensions to the machine learning algorithms for training. We are able to do so because the neural network algorithm handles multi-dimensional data with different types of correlations well. Intuitively, the dimensions represent a mix of different frequencies at a time point, which may be correlated to the current mixture of workloads running in the data centers. We expect the learning algorithm to capture the correlations and thus improve prediction results. Assume the raw time series data of total incoming traffic is $(in^1, in^2, \ldots, in^n)$. Given a time point t and length $l (l \ll n)$, we decompose the time series data $s = (in^{t-l+1}, in^{t-l+2}, \ldots, in^t)$ using Db4 into $(w+1)$ series

$$s_1 = \left(in_1^{t-l+1}, in_1^{t-l+2}, \ldots, in_1^t\right)$$
$$s_2 = \left(in_2^{t-l+1}, in_2^{t-l+2}, \ldots, in_2^t\right)$$
$$\vdots$$
$$s_w = \left(in_w^{t-l+1}, in_w^{t-l+2}, \ldots, in_w^t\right)$$

Then we choose $(in_1^t, in_2^t, \ldots, in_w^t)$ as new features of time point t. Note that the relationship between in^t and the new features is

$$in^t = \sum_{i=1}^{w} in_i^t.$$

The new features can be regarded as decomposition of the raw value and contain the relationship information between the value and the old values. As we can see, each raw dimension is decomposed into w dimensions. Now we have $(2+2M+2L)$ time series data, where M is the number of applications generating elephant flows and L is the number of sublinks related with the data center. With decomposition, we get $(2+2M+2L)\times(w+1)$ - dimensional features for each time point. We denote the new features by f_i. We can choose the parameter l heuristically. In our experiment, we find that $l = 60$, or using 30 minutes of data for decomposition, provides good results.

We then normalize the data before training. The goal of normalization is to scale the data to a given bound. Data normalization can help the learning algorithms avoid computational problems and facilitate network learning [19]. We use z-score [39] to standardize the series data. The z-score is defined as

$$z = \frac{x-\mu}{\sigma}$$

where x is the raw data to be scaled, μ is the mean of dataset and σ is the standard deviation of the dataset. The same μ and σ used to normalized training data are also used to normalize new inputs for prediction.

D. Prediction

We train the normalized data with Artificial Neural Networks (ANNs). ANNs are inspired by biological neural networks. Generally, an ANN consists of multiple layers, including an input layer, a number of hidden layers and an output layer. ANNs can capture non-linear characteristics and find complex relationships between inputs and outputs. ANNs are widely used in function approximation, classification, data processing and robotics [37]. The architecture (e.g., the number of layers, the number of nodes in each layer and so on) of an ANN and optimization algorithms used can affect the final training results.

As usual, we need to specify features and labels for training. Without loss of generality, a data item can be represented as $d_i = (\boldsymbol{f}_i, \boldsymbol{l}_i)$, where $\boldsymbol{f}_i$ stands for the feature vector while $\boldsymbol{l}_i$ stands for the label vector. Usually, we first train a dataset to get a predict function. Then we can predict the labels $(\boldsymbol{l}_i)$

with the function and the features ($\boldsymbol{f}_i$). As mentioned above, by decomposing the total traffic data and the traffic data of elephant flows, we get $(2+2M+2L)\times w$ new features, denoted by f_i, for each time point. Obviously, we should use previous data to predict in^i and out^i. Assume we use the data of k previous steps for one-step-ahead, then we have

$$\boldsymbol{f}_i = \left[f_{i-k+1}, f_{i-k+2}, \ldots, f_i\right]$$
$$\boldsymbol{l}_i = \left[in^{i+1}, out^{i+1}\right]$$

As to multiple-step-ahead prediction, we just need to replace each element of $\boldsymbol{l}_i$ with the corresponding one (e.g., $[in^{i+2}, out^{i+2}]$ for two-step-ahead prediction and $[in^{i+s}, out^{i+s}]$ for s-step-ahead prediction). Thus $\boldsymbol{f}_i$ is a vector of length $(2+2M+2L)\times w\times k$. This means that when we get a predict function F, we pass the $(2+2M+2L)\times w\times k$ features derived from the k previous steps as input to F and get the predicted in^i and out^i.

E. Measure Prediction Errors

We use Relative Root-Mean-Squared Error (RRMSE) to measure prediction errors. It is calculated as follows:

$$RRMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{\theta}_i - \theta_i}{\theta_i}\right)^2},$$

where $\hat{\theta}_i$ is the predicted value and θ_i is the raw value. We can see that it is unitless and can reflect variance and bias at the same time [38].

IV. Experimental Results

We first describe the dataset we use in the evaluation. Then we show that we can achieve significant prediction error reduction over existing methods. Finally we provide details on the effects of different methods and parameters in our prediction model.

A. Experiment Setup

We collect the inter-DC network traffic data of multiple connected production data centers from Baidu for six weeks and our goal is to predict the total incoming/outgoing traffic of a specific data center. The total incoming/outgoing traffic data are direct snapshots of the counters on the edge routers of the specific data center using SNMP, and we collect traffic data for both directions every 30 seconds. We use the data of the last day for testing and the rest for training. Figure 4 shows the total incoming and outgoing traffic of the data center whose total incoming/outgoing traffic is to be predicted. Due to confidentiality concerns, we normalize the Y-axis of all figures so we do not reveal the actual amount of data transfers. The normalization does not affect the results of this paper. For the same reason, as we discuss in the previous section, we present the prediction errors using *relative* root-mean-square-error (RRMSE), instead of RMSE directly.

We use tags, such as source and destination IPs, port, protocol ids, type of service and input/output interface, to identify a flow. We collect the number of packets each flow contributes

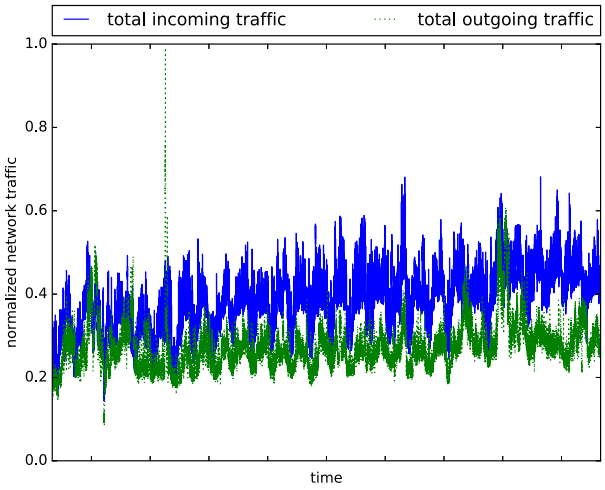

Fig. 4. The total incoming/outgoing traffic.

during a certain period of time and then calculate the average traffic. We sample the flow statistics every five minutes (comparing to the 30 seconds sampling rate for the total traffic) due to the limit of computation and storage resource. We observe the distribution of the traffic and see that the elephant flows from the top-5 applications dominate the traffic, which account for about 80% of the total traffic. In our data center, which runs a few large applications, we only observe these five applications contributing to the vast majority of traffic. Of course, we can add more flows to the elephant flow list, but we feel that 80% is good enough to capture the impact of the large applications on the overall traffic. Figure 5 shows the total traffic of elephant flows we choose. We can see that the traffic of the chosen elephant flows displays a substantial, but not perfect correlation with the traffic of the total flows. The production data center connects with 4 other data centers, so we collect the traffic data of 4 sublinks. Note that the summation of incoming/outgoing traffic of the sublinks equals the total incoming/outgoing traffic of the production data center.

In our experiment, we use one day data as test data. As we take a sample every 30 seconds, there are 2880 values we are predicting for the day. We then perform 30-second-ahead, 1-minute-ahead, 5-minute-ahead, 10-minute-ahead, 15-minute-ahead and 20-minute-ahead prediction and compare the differences between among strategies in each case.

B. Overall Prediction Measurement

We reduce the prediction error by performing wavelet transformation, and adding interpolated traffic data of elephant flows and sublinks. We also put the total incoming and outgoing traffic together for training, thus we can use one model to predict both the incoming and outgoing traffic. We then found a suitable training set size.

We compare our model with two well-known models: one is a representative traditional linear model ARIMA [20], the other is ANN without wavelet transform and interpolation, as many existing works do [6], [19].

In the following evaluation, we use one input layer, one hidden layer and one output layer for the artificial neural network. We also evaluate the prediction accuracy using more

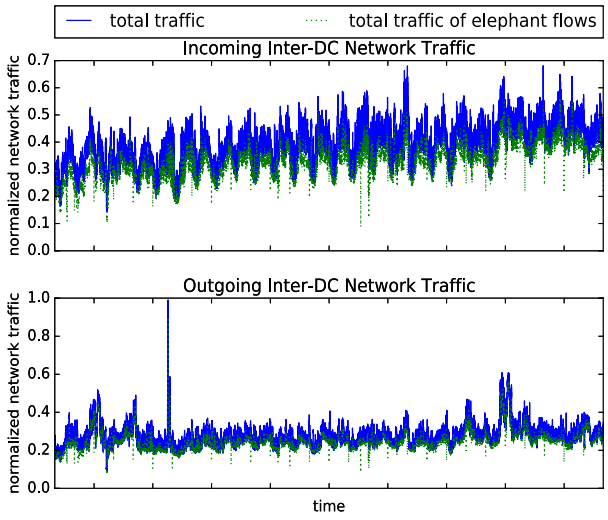

Fig. 5. Correlations between total traffic and the total traffic of the elephant flows. The elephant flows occupy a large portion of the total traffic.

TABLE I
PREDICTION ERRORS (RRMSE) FOR INCOMING TRAFFIC

	30s	1min	5min	10min	15min	20min
ANN	0.0439	0.0525	0.080	0.096	0.105	0.113
ARIMA	0.0398	0.0496	0.0793	0.0971	0.111	0.119
Ours	0.0305	0.0444	0.0751	0.0901	0.0993	0.107

TABLE II
PREDICTION ERRORS (RRMSE) FOR OUTGOING TRAFFIC

	30s	1min	5min	10min	15min	20min
ANN	0.0439	0.0522	0.0808	0.0967	0.1089	0.117
ARIMA	0.0396	0.0492	0.0795	0.0980	0.112	0.122
Ours	0.0306	0.0444	0.0766	0.0909	0.102	0.109

than one hidden layers and did not find much difference. We use Stochastic Gradient Descent (SGD) [11] as the optimization algorithm for model training. To include elephant flow data, we use zero interpolation method. We will evaluate other interpolation methods in the next section.

Table I, II and Figure 6 show the comparison results. We show that for 30-second-ahead prediction, our model reduces the prediction errors by about 30% for incoming/outgoing traffic compared to the linear model and 23% compared to the conventional ANN. Also, for 1-minute-ahead prediction and longer-time-ahead prediction, our model reduces the prediction errors by 5%∼15% for incoming/outgoing traffic compared to the linear model and conventional ANN.

The accuracy improvement is essential for production: using the improved prediction results as guidance for traffic scheduling, Baidu is able to reduce the peak inter-DC link utilization (the ISP's billed utilization) by about 9%. The actual implementation and the evaluation of the prediction-based traffic scheduling system is out of the scope of the paper and thus omitted here.

C. Effect of Different Factors in Our Model

The prediction error reduction is the result of a combination of different methods and parameters. We evaluate the effects of the key components in our model.

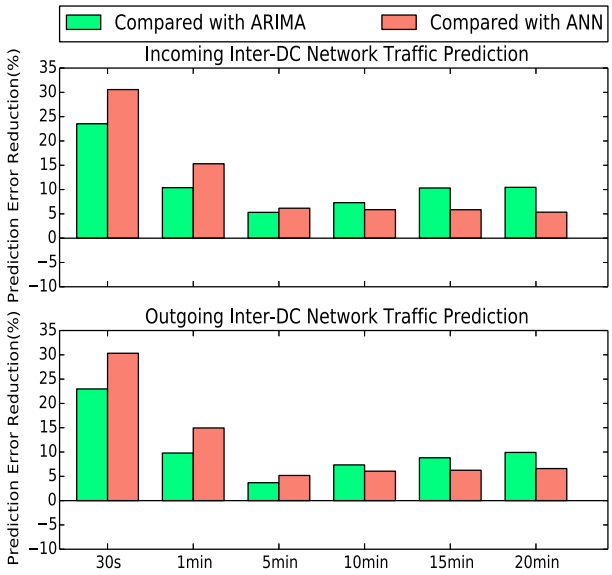

Fig. 6. Prediction error reduction over ARIMA and ANN. Positive numbers mean that we reduce the prediction errors actually while negative numbers mean the opposite. We can see that our model reduces prediction errors significantly for long-term-ahead prediction.

1) Length of Training Set: Intuitively, using longer history as training set can help reducing the data noise and thus prediction errors can be reduced to a certain point. A large training set may be of little use while bringing in extra and unnecessary training cost. Our evaluation confirms this intuition.

Thus we need to balance the advantages with the disadvantages of increasing the training set size. We compare the performance of different training set sizes. Figure 7 shows that using a training history of longer than 4 weeks, we can obtain a good enough model. We are still evaluating if it is related to a regular monthly pattern, collecting data for a much longer term, which is an important future work for us.

2) Effectiveness of Wavelet Transform: We use Daubechie's 4 (Db4) wavelets with ten levels of decomposition, as [23] did. For each time point, we decompose the subseries consisting of 60 values (including the current one) to get 11 new feature as Section III-C describes. We use a 4-week history for training. Figure 8 compares the prediction errors with and without wavelet transform.

Wavelet transform is an essential preprocessing step: for different steps prediction, the wavelet transform reduces the average prediction errors by 5.4% and 2.9% for incoming and outgoing traffic. Intuitively, learning methods like ANNs work because they capture the (non-linear) correlations among multiple dimensions of data. Wavelet transform adds dimensions representing the reoccurring patterns of the data and reveals another level of important correlations. Thus the combination of wavelet transform and ANN brings a satisfying prediction error reduction.

3) Combining Incoming/Outgoing Traffic in the Same Model: As we discussed in Section III-A, we can either train separate models for incoming and outgoing traffic, or we can combine both traffic numbers into the same model. This is a key benefit of using learning methods like ANN – we have the flexibility to combine prediction models without changing

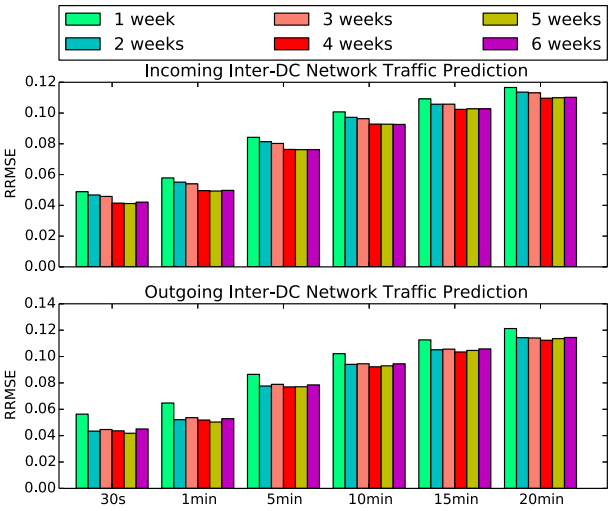

Fig. 7. Prediction errors of the models with different training set sizes.

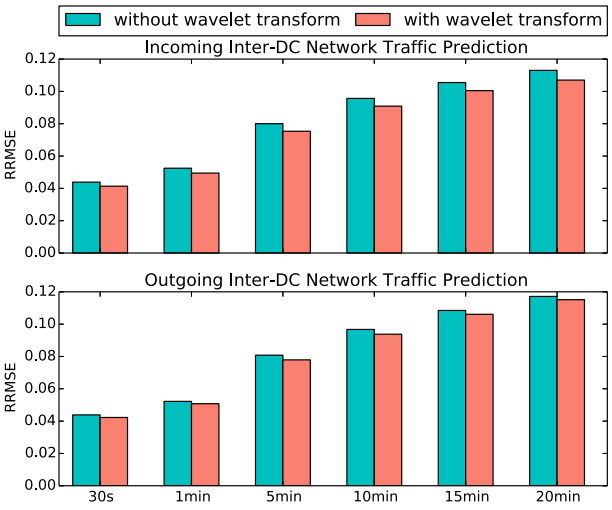

Fig. 8. The reduction in prediction errors with/without wavelet transform.

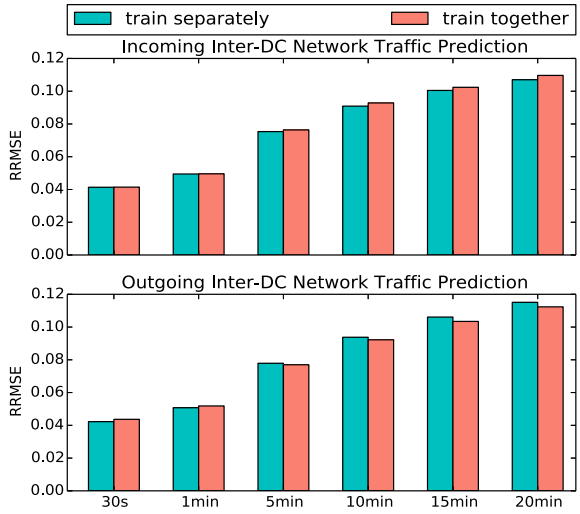

Fig. 9. Prediction errors: combining incoming flows and outgoing flows vs. training separate model for them. We can see that they provide similar results. Giving that the combined model trains faster, we use the combined model.

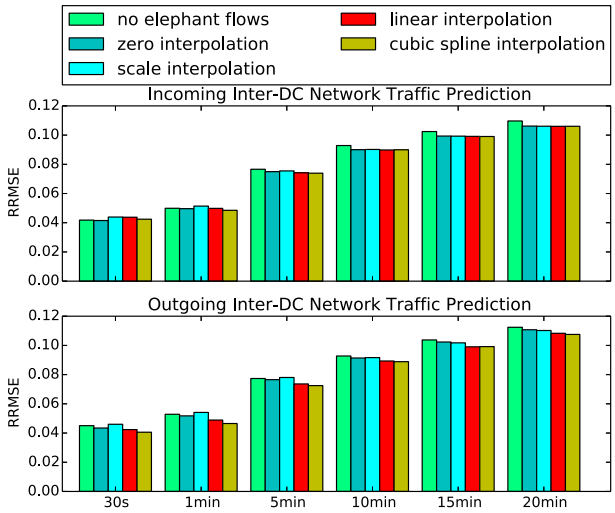

Fig. 10. Prediction errors of the models with/without elephant flows using different kinds of interpolations.

to the model itself. Here we compare the result of these two alternatives.

Figure 9 shows the comparison results. There is no significant difference in prediction accuracy. This is as expected because the incoming and outgoing traffic are highly correlated.

It is beneficial to use the combined model. The single model is not only easier to implement and maintain, but also it needs less time to train comparing to the two separate models. In our experiments, using the combined model approach reduces the training time by about 40% comparing with the separate models.

4) Elephant Flows: The elephant flows play an important role in our model. Figure 10 shows the results of adding elephant flow information using different interpolation methods. We have the following observations from the figure.

First, elephant flow information reduces the prediction errors. For both incoming and outgoing traffic, adding elephant flows information reduces prediction errors, especially for the 5-minute or longer time ahead prediction.

Second, different interpolation methods have similar effects, except for the scale interpolation. As we have discussed in Section III-B, as ANN works on the correlations among different dimensions, the assumed correlation between the traffic of elephant flows and total traffic actually negatively affects the power of ANN. Interpolation methods that consider the neighbor values (e.g., the linear or cubic interpolation) perform slightly better than zero interpolation, which is as expected. Given the good balance between simplicity and performance of zero interpolation, we choose it as our interpolation method in production.

Third, we observe that the more accurate number of elephant flow is, whether the measurement comes from interpolation or actual measurements from flow sampling, the better the overall prediction accuracy is.

Intuitively, the wavelet transform and ANN capture all the recurring patterns of the total traffic, but the elephant flows contribute to the overall traffic in a much more random way.

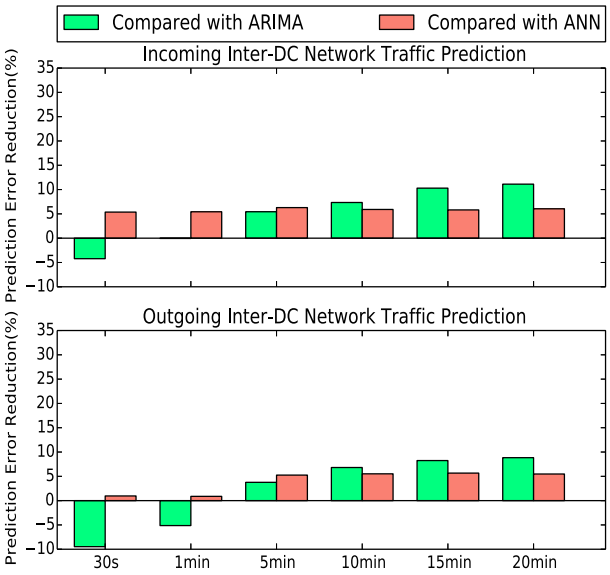

Fig. 11. Prediction errors of the models with elephant flows (but without sublink traffic information).

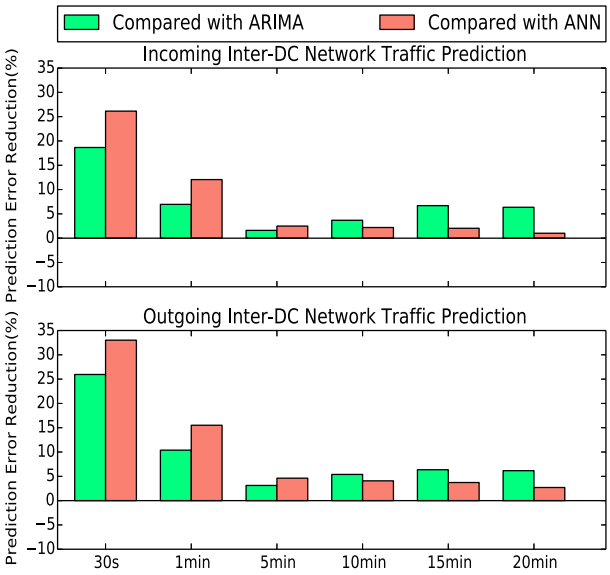

Fig. 12. Prediction errors of the models with/without sublink traffic information.

We use the traffic data of elephant flows to "calibrate" the total traffic prediction, and thus the accuracy of elephant flows plays an important role. As an on-going future work, we are improving our elephant flow monitoring system to provide more frequent measurements.

We compare the prediction errors of the models using these four kinds of interpolations, as Fig. 10 shows. We can see that cubic spline interpolation performs the best. Linear interpolation performs similar with cubic spline interpolation, especially for long-term-ahead prediction. Zero interpolation performs a little worse than the two pervious ones, but performs better than scale interpolation. Scale interpolation performs worst among the four kinds of interpolations. For incoming inter-DC network prediction, the model without elephant flows performs a little better than the models with elephant flows for 30-second-ahead prediction (namely one-step-ahead here), but performs worse for 1-minute-ahead or longer time ahead prediction. For outgoing inter-DC network prediction, the model without elephant flows performs worse than most of the models with elephant flows, especially for long-term-ahead prediction. The reason why zero interpolation and scale interpolation perform a little worse than linear interpolation and cubic spline interpolation may be that, each traffic value is not independent with the previous and next ones, but zero interpolation and scale interpolation use only the previous values to interpolate new values while the other two methods consider both the previous and future ones.

However, as we know, when we perform linear interpolation at x between x_i and x_{i+1}, we should know x_i and x_{i+1} in advance. Cubic spline interpolation even require that all the points should be known in advance (see Section III). In practice, we have no way to know the future traffic so that we cannot perform such interpolations correctly while trying to construct the missing values of the traffic of the elephant flows for prediction. On the other hand, interpolations like zero interpolation and scale interpolation don't require this. As we can see, zero interpolation performs better than scale interpolation, we use zero interpolation in real-world applications.

As the better interpolation methods here (i.e., linear interpolation and cubic spline interpolation) take the dependency of missing traffic values and previous and future ones, the interpolated values using these methods may be closer to the real ones. Besides, we found that the average prediction errors at the time point where the traffic data of elephant flows were sampled were a little smaller than the average prediction errors at the time point where the traffic data of elephant flows were interpolated. Thus, we can say that, the more frequently we sample the elephant flows, the more real traffic data we get, the smaller the bias between the constructed values and the real values is and the better results we can get.

Figure 11 shows the comparison results between ANN and ARIMA and our model with elephant flows. From the figure, we can see that our model with elephant flows (without sublinks) performs worse than convential methods for 30-second-ahead and 1-minute-ahead prediction but better for 5-minute-ahead or longer-ahead prediction.

5) Sublink Traffic: Adding explicitly the sublink traffic information to our model is as important as adding the elephant flows. To see the effectiveness of adding traffic data of sublinks, we first remove traffic data of elephant flows and test our model with sublink traffic. Figure 12 shows the comparison results between ANN and ARIMA and our model with sublink traffic (but without elephant flows). From the figure, we can see that our model with sublink traffic performs pretty better than convential methods for 30-second-ahead and 1-minute-ahead prediction and a little better for 5-minute-ahead or longer-time-ahead prediction. Meanwhile, we should notice that for 5-minute-ahead or longer-time-ahead prediction, our model with sublink traffic information performs a little worse than that with elephant flows. In other words,

elephant flows help improve the prediction accuracy of long-term-ahead prediction while we benefit more from sublink traffic for short-term-ahead prediction. That is, sublink traffic and elephant flows are complementary.

Why does this happen? One possible explanation is that the elephant flows, such as the flows generated by Hadoop, always last for a relatively long time but may fluctuate significantly during their lifetime, so elephant flows help long-term-ahead prediction a lot but may interfere short-term-ahead prediction. On the other hand, traffic of each sublink consists of many flows generated by different jobs, thus the statistical characteristics are relatively stable and will not be affected by the changes of a fraction of jobs in a short time. Thus the sublink traffic data can help more for short-term-ahead prediction. Based on this, we decide to include both elephant flows and sublink traffic information in our model to help improving the prediction accuracy, as Figure 6 shows, from which we can see that both the short-term-ahead and long-term-ahead prediction errors are reduced significantly.

V. Conclusion

We propose a new model for inter-DC network traffic prediction. In contrast with normal network traffic, inter-DC traffic are dominated by a few large applications producing elephant flows and the combination of sublink traffic further complicates the issue. We can view the traffic as a combination of reoccurring patterns and some large noise.

The key for the traffic prediction is decomposing the various components from the combined traffic pattern. We decompose the traffic in two ways: first we use Db4 wavelet transform to decompose the time domain traffic data. Then we also add explicit information about elephant flows and sublink traffic. The elephant flow information provides multiple calibration points that significantly reduce the prediction errors, especially for 5-minute-ahead or longer-time-ahead prediction. Traffic of sublinks, on the other hand, helps more for 30-second-head and 1-minute-ahead prediction. We show that using the combination of wavelet transform, elephant flows and sublink traffic data, we can reduce the prediction errors significantly.

We emphasize on practical issues in the prediction model design, especially the cost of measurements. We show that we can significantly reduce the flow sampling overhead using interpolation methods. We also evaluate the possibility of reducing the training overhead by combining both incoming and outgoing traffic into the same model, reducing the training overhead by 40%. Our prediction method can help Baidu reduce the peak bandwidth for about 9% on average. The monetary cost reduction is significant for large scale inter-DC network. Thus the accuracy improvement is necessary and worthwhile.

As future work, we will extend the prediction to a longer time periods (weeks to months) to support tasks like resource provisioning. We will also explore models to predict the traffic on core switches within a data center. On the engineering side, we are improving the technique to elephant flows traffic at a higher frequency.

References

[1] H. Wang *et al*., "COPE: Traffic engineering in dynamic networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 99–110, 2006.

[2] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. 7th Conf. Emerg. Netw. Exp. Technol.*, Tokyo, Japan, 2011, Art. no. 8.

[3] C.-Y. Hong *et al*., "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–26, 2013.

[4] S. Jain *et al*., "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[5] S. L. Ho, M. Xie, and T. N. Goh, "A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction," *Comput. Ind. Eng.*, vol. 42, nos. 2–4, pp. 371–375, 2002.

[6] W. Cheng and P. Feng, "Network traffic prediction algorithm research based on PSO-BP neural network," in *Proc. Int. Conf. Intell. Syst. Res. Mechatron. Eng.*, Zhengzhou, China, 2015. [Online]. Available: http://www.atlantis-press.com/php/pub.php?publication=isrme-15

[7] H. Feng and Y. Shu, "Study on network traffic prediction techniques," in *Proc. Int. Conf. Wireless Commun. Netw. Mobile Comput.*, Wuhan, China, 2005, pp. 1041–1044.

[8] K. Papagiannaki *et al*., "A pragmatic definition of elephants in Internet backbone traffic," in *Proc. 2nd ACM SIGCOMM Workshop Internet Meas.*, Marseilles, France, 2002, pp. 175–176.

[9] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[10] W. Yoo and A. Sim, "Network bandwidth utilization forecast model on high bandwidth networks," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, Garden Grove, CA, USA, 2015, pp. 494–498.

[11] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT'2010*. Heidelberg, Germany: Physica-Verlag HD, 2010, pp. 177–186.

[12] K. Zhang, Y. Chai, and X.-A. Fu, "A network traffic prediction model based on recurrent wavelet neural network," in *Proc. 2nd Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, Changchun, China, 2012, pp. 1630–1633.

[13] V. Alarcon-Aquino and J. A. Barria, "Multiresolution FIR neural-network-based learning algorithm applied to network traffic prediction," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 2, pp. 208–220, Mar. 2006.

[14] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, Jan. 2003.

[15] B. Yegnanarayana, *Artificial Neural Networks*. New Delhi, India: PHI Learn. Pvt. Ltd., 2009.

[16] J. Liu and Y.-L. Huang, "Nonlinear network traffic prediction based on BP neural network," *J. Comput. Appl.*, vol. 7, no. 7, pp. 1770–1772, 2007.

[17] H. Akaike, "Fitting autoregressive models for prediction," *Ann. Inst. Stat. Math.*, vol. 21, no. 1, pp. 243–247, 1969.

[18] J. L. Torres, A. García, M. De Blas, and A. De Francisco, "Forecast of hourly average wind speed with ARMA models in Navarre (Spain)," *Sol. Energy*, vol. 79, no. 1, pp. 65–77, 2005.

[19] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[20] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Trans. Power Syst.*, vol. 18, no. 3, pp. 1014–1020, Aug. 2003.

[21] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A seasonal-trend decomposition procedure based on loess," *J. Off. Stat.*, vol. 6, no. 1, pp. 3–73, 1990.

[22] C. S. Burrus, R. A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.

[23] H. He and J. A. Starzyk, "A self-organizing learning array system for power quality classification based on wavelet transform," *IEEE Trans. Power Del.*, vol. 21, no. 1, pp. 286–295, Jan. 2006.

[24] T. Mori, R. Kawahara, S. Naito, and S. Goto, "On the characteristics of Internet traffic variability: Spikes and elephants," *IEICE Trans. Inf. Syst.*, vol. 87, no. 12, pp. 2644–2653, Feb. 2004.

[25] C. De Boor, "A practical guide to splines," in *Mathematics of Computation*. New York, NY, USA: Springer, 1978.

[26] K. Hu, A. Sim, D. Antoniades, and C. Dovrolis, "Estimating and forecasting network traffic performance based on statistical patterns observed in SNMP data," in *Machine Learning and Data Mining in Pattern Recognition*. Heidelberg, Germany: Springer, 2013, pp. 601–615.
[27] B. Zhou, D. He, Z. Sun, and W. H. Ng, "Network traffic modeling and prediction with ARIMA/GARCH," in *Proc. HET-NETs Conf.*, 2005, pp. 1–10.
[28] C. W. J. Granger and R. Joyeux, "An introduction to long-memory time series models and fractional differencing," *J. Time Series Anal.*, vol. 1, no. 1, pp. 15–29, 1980.
[29] S. Periyanayagi and V. Sumathy, "S-ARMA model for network traffic prediction in wireless sensor networks," *J. Theor. Appl. Inf. Technol.*, vol. 60, no. 3, pp. 524–530, 2014.
[30] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk, "A multifractal wavelet model with application to network traffic," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 992–1018, Apr. 1999.
[31] H. Zhao and N. Ansari, "Wavelet transform based network traffic prediction: A fast on-line approach," *J. Comput. Inf. Technol.*, vol. 20, no. 1, pp. 15–25, 2012.
[32] X. Tan, W. Fang, and Y. Qu, "Network traffic prediction algorithm based on wavelet transform," *Int. J. Adv. Comput. Technol.*, vol. 5, no. 5, p. 183, 2013.
[33] H. Feng, Y. Shu, S. Wang, and M. Ma, "SVM-based models for predicting WLAN traffic," in *Proc. IEEE Int. Conf. Commun. ICC*, vol. 2. Istanbul, Turkey, 2006, pp. 597–602.
[34] Y. Qian, J. Xia, K. Fu, and R. Zhang, "Network traffic forecasting by support vector machines based on empirical mode decomposition denoising," in *Proc. 2nd Int. Conf. Consum. Electron. Commun. Netw. (CECNet)*, Yichang, China, 2012, pp. 3327–3330.
[35] Z. Wu and N. E. Huang, "Ensemble empirical mode decomposition: A noise-assisted data analysis method," *Adv. Adap. Data Anal.*, vol. 1, no. 1, pp. 1–41, 2009.
[36] H. Xiao, H. Sun, B. Ran, and Y. Oh, "Fuzzy-neural network traffic prediction framework with wavelet decomposition," *Transp. Res. Rec. J. Transp. Res. Board*, vol. 1836, no. 1, pp. 16–20, 2003.
[37] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*, vol. 20. Boston, MA, USA: PWS, 1996.
[38] D. Mouillot and A. Leprêtre, "A comparison of species diversity estimators," *Res. Popul. Ecol.*, vol. 41, no. 2, pp. 203–215, 1999.
[39] E. Kreyszig, *Applied Mathematics*. New York, NY, USA: Wiley, 1979.

Yi Li (S'16) received the B.S. degree from Peking University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing. His current research interest includes network management, big data analysis, cloud computing and privacy.

Hong Liu received the M.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014. He has been a Senior Research and Development Engineer with Baidu Inc., since 2014. His interests lie in statistical machine learning and deep learning.

Wenjun Yang received the M.S. degree from Tsinghua University, Beijing, China, in 2013. He is with Didi Inc., as a Senior Engineer. His interests lie in data mining and computer architecture.

Dianming Hu has been a Principal Engineer and the Director of the Cloud and Embedded Platform Group, NovuMind Inc., Beijing, China, since 2016. He was the Leader of Baidu's Datacenter Intelligence Research and Development Team for the past five years. His main fields of research include device intelligence, cloud, and datacenter system.

Xiaojing Wang received the M.S. degree from the Department of Computer Science and Technology, Northeastern University, Shenyang, China, in 2008. She is currently a Senior Project Manager with the Data Center Intelligence Team, Baidu Inc., Beijing, China. She and her team are currently working on decision optimizing in data center with big data and machine learning.

Wei Xu received the B.S. degree from the University of Pennsylvania in 2003 and the Ph.D. degree in computer science from the University of California at Berkeley in 2010. He is an Assistant Professor with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. He is the Director of Open Compute Project Certification Laboratory, China. He was a Software Engineer with Google working on logging and debugging. He has broad research interests in distributed system design, big data, data center networking, system management and debugging, large scale system for machine learning and data mining, as well as various big data applications. He was a recipient of the National Youth 1000 Program of China in 2013.