A Power Consumption Model for Cloud Servers Based on Elman Neural Network

Wentai Wu, Weiwei Lin, Ligang He, *Member, IEEE,* Guangxin Wu, and Ching-Hsien Hsu, *Senior Member, IEEE*

Abstract — Leveraging power consumption models in software systems can achieve easy deployment of low-cost, high-availability power monitoring in cloud datacenters that are usually large-scale, heterogeneous and frequently scaling up. However, traditional regression-based power consumption models generally have two drawbacks. First, their mathematical forms are usually fixed and determined a priori. This may cause unacceptable increase of error or over-fitting as the power signatures of cloud servers are usually uncertain. Second, the characteristic of workload dispatched to cloud servers is constantly changing while regression-based models can hardly generalize to a wide range of servers and workload types. As a novel solution, we in this paper propose a server power consumption model based on Elman Neural Network (PCM-ENN), aiming to allow accurate and flexible power estimation. PCM-ENN is an end-to-end black box model capable of learning the temporal relation between samples in a time series of power consumption. We trained and evaluated PCM-ENN on two power sequence datasets collected from heterogeneous hardware and operating systems running quasi-production benchmarks like CloudSuite. Experimental result shows that PCM-ENN generated accurate estimates on server power consumption with only small errors, outperforming widely-used linear regression model and NARX model in terms of accuracy.

Index Terms — Cloud servers; Cloud datacenters; Power time series; Power consumption models; Elman neural network

1 Introduction

WHILE cloud computing is still gaining increasing popularity around the world, excessive electricity consumption by cloud datacenters has become a prominent issue and drawn a lot of concern. Statistics shows that the annual electricity consumed by datacenters in the USA already reached 91 billion kilowatt-hours, while the figure is projected to soar to as high as 140 billion kWh in 2020 [1]. Over-consumption of energy certainly makes negative impacts on the development of cloud computing, bringing about problems such as increasing operation cost and adverse effects on environment.

Implementation of fine-grained power monitoring systems is the very foundation for realizing energy-aware power provisioning and management. Emerson's report in North America reveals that 51% of respondents cited adequate monitoring/ datacenter management capabilities among their three biggest concerns [2]. Traditionally, server power is measured using external metering devices or dedicated data acquisition interfaces. For instance, IBM PowerExecutive [30] is a plug-in tool for gaining as well as capping actual power consumption of servers under the specified architecture – System X. Hardware (e.g., sensors) power measuring can be the best option for homogeneous datacenters but is hardly a solution for heterogeneous ones such as legacy systems and cloud datacenters [28][29]. The main reasons are but not limited to expensiveness, poor scalability [3] and coarse granularity. By contrast, power monitoring systems built on software are able to support fine-grained, low-cost, easy-to-extend monitoring in a cloud system that can be highly heterogeneous and constantly scaling [29]. The core of software power monitoring is the pre-built power consumption model, which is defined as one or multiple functions that map system performance related metrics to system power or energy consumption [4]. Power consumption model takes as input one or several metrics (features) at different sampling granularities (e.g., OS level and processor level), outputs estimated values of power (for an instant) or energy consumption (for a period). Power model is not only used for monitoring purposes, but also provides important guidance for energy-aware resource provisioning [5][39][41], capping [38] and scheduling [6][7][40]. For example, Shen et al. [38] proposed "power container", which is a novel operating system facility that accounts for and controls the power and energy usage of every single task on multi-core systems. One of the key techniques in their work is an online, adaptive power model for capturing the power consumption of concurrent tasks. Niu et al. [8] implemented an energy-aware scheduling framework named GreenMR. The authors introduce the execution time models and power consumption models for Map phase and Reduce phase, separately. The models are used to profile the total

- *W. Wu and L. He are with the Department of Computer Science, University of Warwick, Coventry, CV4 7AL, United Kingdom. E-mail: {wentai.wu, ligang.he}@warwick.ac.uk.*
- *W. Lin (corresponding author) is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: linww@scut.edu.cn.*
- *W. Wu and G. Wu are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: {cswuwt, cswgx1nfinite}@mail.scut.edu.cn.*
- *C.-H. Hsu is with the Department of Computer Science and Information Engineering, Chung Hua University, 707, Sec. 2, WuFu Rd, Hsinchu 300, Taiwan. E-mail: chh@cs.ccu.edu.tw.*

run time and energy consumption jobs in a given resource configuration.

Previous studies mainly used regression-based methods to build power consumption models. The most widely adopted approach is linear regression because of its good interpretability and simplicity in training [9]. For example, Hsu and Poole [10] investigated a number of regression-based power consumption models which are all functions of CPU utilization. Lin et al. [11] surveyed mainstream component-level power models and evaluated their accuracy in experiment using regression analysis. Whereas regression models are commonly adopted, they have outstanding deficiencies. First, using fixed forms limits their ability to generalize to a diversity of server power curves. Second, they can hardly support incremental training whilst cloud infrastructures are upgrading fast. Third, temporal relation between time-series records (samples of power consumption in a time series) is neglected in regression models.

More and more studies on time-series processing begin to adopt Artificial Neural Network (ANN), but most of them focus on predicting power or workload in the future. For example, we have seen promising results in predicting workload using ANN or its improved forms [12][13][14]. ANN is complex but allows flexible training and more possibilities in model optimization, providing an entirely different way to build and train power consumption models that are suitable for cloud servers. In this paper, we propose to leverage artificial neural network and find the proper form of it to build the power consumption model for cloud servers. Fig. 1 demonstrates a software power monitoring framework applied to a cloud system where power models are applied to heterogeneous cloud servers. The models are trained on historical datasets collected from corresponding types of cloud servers.

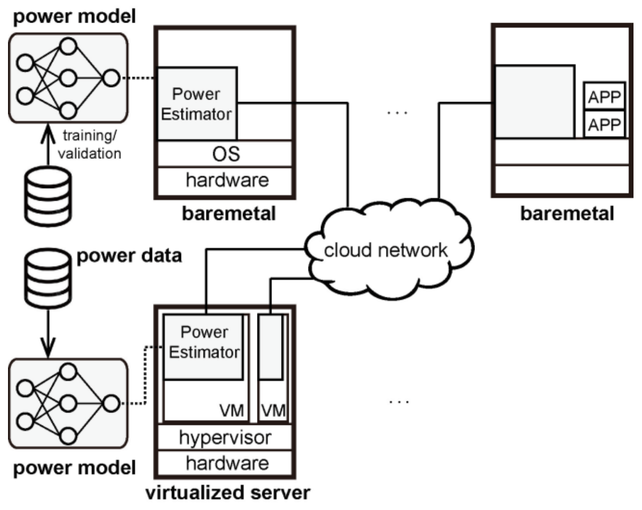


Fig. 1. The proposed framework of power monitoring using power consumption models in cloud datacenters.

Training data for each model is server-specific, whereas different servers with identical hardware can share a same power consumption model, which largely reduces the number of models needed. In particular, we take advantage of neural network to improve power models' accuracy, enable incremental training, and enhance their ability to generalize what it learned. Moreover, we in this paper explore the temporal correlation between consecutive power records using a simple recurrent structure. We summarize the main contributions of our work as follows:

1. Based on the fundamental requirements of a power monitoring system, we first introduce a number of ANN architectures that can be applied to power estimation after training on time-series data set. We further summarize their advantages and limitations.
2. We propose a server power consumption model based on Elman Neural Network (PCM-ENN), which is able to learn the temporal impact from previous power consumption and make real-time estimation.
3. We trained PCM-ENN on mixed datasets containing multiple types (CPU-intensive, memory-intensive and I/O-intensive) of workload, and evaluated it on test datasets obtained by running a production benchmark suite. Experimental result on two completely heterogeneous servers shows that PCM-ENN is more accurate than linear regression model, NARX model and the monitoring software Joulemeter.

The rest of this paper is organized as follows. Section 2 introduces related work on prediction models and power models based on ANN. Section 3 summarizes a number of ANN structures that can be applied to processing power sequence. We present the proposed server power model based on ENN in section 4 and show evaluation results in section 5. Finally we conclude the paper in section 6.

2 RELATED WORK

Lin et al. [3] in their paper categorize power measuring methods into four classes: direct metering by hardware devices, power estimation using power consumption models, power measuring in virtualized environment, and simulation-based power estimation. They also figure out that traditional power metering with external devices or dedicated acquisition systems (e.g., IBM Active Energy Manager [15]) is not feasible in large-scale and heterogeneous datacenters mainly due to the problem of hardware compatibility.

Software monitoring systems built on power consumption model has the advantages of low deployment cost, high scalability and fine granularity [28], and they are also applicable to estimating the power of virtual machines [42] and containers. For instance, Joulemeter [16], a power monitoring tool developed by Microsoft, works on the basis of several component power models. The widely-used simulation framework CloudSim [17] uses pre-defined models to emulate cloud servers' power consumption. Software-based accounting can also be applied to Non-IT units at the granularity of virtual machine [43]. Dayarathna et al. [18] in their paper discuss power models at different levels of granularity from single components to a whole datacenter entity, and figure out that a well-designed model should be six features including accuracy, speed, generality, portability, inexpensiveness, and simplicity. Based on component-level power models, Lin et al. [19] developed a distributed power consumption monitoring tool named EnergyMeter. Their work adopts a white box approach of power modeling and is based on a multi-component system model. They proposed to use three separated, independent component power models to estimate CPU, memory and disk's power, and combine them with the system idle power to obtain the estimate of the whole computer's power. Similar to Joulemeter, the solution is light-weight (because it only needs to track utilizations) and provides fine-grained power monitoring. However, a disadvantage of it is that multiple power models corresponding to multiple components have to be maintained. One or even all of the models may need to be retrained in case system hardware is upgraded. Tang et al. [29] proposed a software-based hierarchical power estimation approach for legacy datacenters. Similar to our rationale, they build power mapping functions (PMFs) of server states along with a selective incremental training process to achieve non-intrusive, zero-cost, accurate power monitoring. The work of Lin et al. [11] summarizes a great number of power models in different forms. They also experimentally evaluated the models and the results show a fact that the most suitable model differs from server to server. In other words, it is unpractical to find a fixed form of power model that fits all types of servers.

Regression models may perform poorly in heterogeneous environments. Dayarathna et al. [18] believed the main causes are the cross dependence between selected features, features' being outdated after hardware upgrade, some features' strong dependency on OS, and the complexity of contemporary system architecture. As an alternative, artificial neural network is attracting more attention. Similar to power monitoring, workload prediction is a significant technique for datacenter management. Many studies have already adopted different ANN structures, like BPNN [20], LSTM-ANN [21] and Fuzzy ANN [22], to build prediction models and demonstrated promising results. Kumar and Singh [12] built a simple feed-forward neural network and trained it with differential evolution algorithm. Prevost et al. [23] used a neural network and an Auto Regression Prediction Weiner Filter to predict cloud datacenters' workload. Results proved that both of the models are accurate. The number of historical data inputs is a critical hyper-parameter, Roy et al [24] carried out experiment and figured out that with only three most recent data records the future workload can be accurately forecasted. Kumar et al. [21] took advantage of LSTM-ANN to predict the number of requests received by web servers. They compared their model with BPNN and showed a significant improvement in accuracy. Chen et al. [22] combined ensemble model and fuzzy neural network to make workload prediction. The fuzzy neural network takes as input the outputs of several base predictors and consists of six layers.

Power estimation and load prediction are different but quite similar in essence. First, both of them are traditionally done by mathematical models and regression analysis, and can both resort to new modeling methods like neural network. Besides, monitoring power consumption and workload will both generate time-series data, which also indicates that we should consider the relation between temporally neighboring data records. Ruiz et al. [25] proposed to use artificial neural network for energy consumption forecasting. They trained three typical neural network models - NAR (Non-linear Auto-Regressive Neural Network), NARX (NAR with exogenous inputs) and Elman Neural Network on a historical data set of buildings' energy use. As a result, their study reveals neural networks' potential in performing accurate power forecasting for buildings. However, how to utilize them to build power estimator for cloud servers need to be further explored.

Accuracy can no longer be guaranteed using traditional power models since the heterogeneity of both servers and workload becomes increasingly common in cloud datacenters. Thus it is of great necessity to explore how to build power models that are easy-to-generalize.

3 Modeling time series of power with ANN

Traditional power consumption models assume that power consumptions at different moments are independent. But as a matter of fact, system power actually changes in a continuous manner (similar to the change of workload), and there is also experimental evidence supporting the implicit relation between power consumption at consecutive moments [25]. Basically, power consumption models can be categorized into two types: (1) predicting power by examining historical data, and (2) estimating current system power through collecting relevant performance metrics such as utilization. The first one can be achieved in a way nearly the same with workload forecasting [20]. We focus on the second type as our goal is to establish real-time power monitoring on cloud servers without any extra metering devices. Li et al. [26] built a software/program power consumption model using BPNN. The model takes as input the target program's time complexity, space complexity and data size, and was experimentally proved accurate. However, BPNN model does not take into account time sequence patterns. Adopting a different approach, we attempted to take advantage of recurrent neural network to build a system-level end-to-end power consumption model. We aim to realize precise, real-time power estimation exploiting commonly-used features that are easy to collect in OS. Considering that power modeling is not of high complexity (as a regression task essentially), we believe a simple recurrent neural network structure with one hidden layer is enough for power estimation.

In this section we introduce a number of ANN structures applicable to power estimation followed by a brief comparison of their strengths and limitations.

3.1 BPNN Models

Back Propagation Neural Network (BPNN) is a commonly used ANN structure that applies error back propagation to model training. Basically, there are two BPNN structures corresponding to power prediction and power estimation, respectively.

The first structure is used to predict power consumption by taking historical power data as input. We in this paper call it Sliding Window BPNN power model. Sliding Window BPNN power model makes prediction entirely based on historical data. This probably leads to poor accuracy because the change of power consumption shows large uncertainty and weak temporal correlation. Moreover, we have to feed historical power records into the model, which means that the Sliding Window BPNN model cannot work without ground truth (i.e., measured data) in the current window. Thus, the model is not fit for power estimation. The second BPNN structure has a different design of input layer, which represents a vector of system performance features such as CPU utilization and disk throughput rate. BPNN model with only features input has a drawback in common with regression model - it cannot learn the temporal relation between consecutive time-series records. But its advantage over regression model is that we can easily enhance its complexity by extending the hidden layer by adding neurons or increasing the number of layers.

3.2 NAR(X) Network Model

Non-linear Auto-Regressive Model (NAR) and Non-linear Auto-Regressive Model with Exogenous Inputs (NARX) are two forms of recurrent neural network commonly used in establishing prediction model on time-series data set.

NAR receives power values at n consecutive moments denoted as $p(t\text{-}n)$, $p(t\text{-}n+1)$, …, $p(t\text{-}1)$. Its output layer typically contains only one neuron corresponding to the power consumption at time t. The limitation of NAR is similar to that of the Sliding Window BPNN model. The reason is that neither of them takes system performance features into account. But NAR takes advantage of the feedback from its output (current prediction) to input (feature input of next moment) to enables the model to work continuously and automatically without feeding of historical data.

NARX, as shown in Fig. 2, extends NAR network by introducing additional input neuron(s) to receive exogenous input(s). For power consumption model, we typically choose system utilization features as exogenous inputs. In Fig. 2, u_d stands for the dth dimension of the input feature vector. NARX's advantage over NAR is that it explicitly associates current system power to both performance features and historical power. However, NARX increases the risk of over-fitting as the size of model input is enlarged.

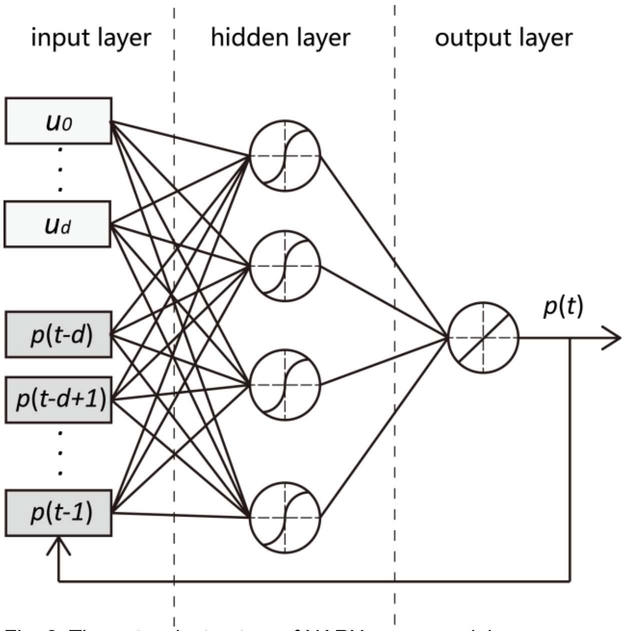


Fig. 2. The network structure of NARX power model

3.3 ENN Model

Elman Neural Network (ENN)[31], proposed by Jeffrey Elman, is also referred to as Simple Recurrent Neural Network. ENN was first applied to automatic speech processing and soon proved effective in a wide range of time-series processing tasks. Different from Jordan Neural Network (e.g. NAR and NARX), the ultimate output of ENN is not directly fed back to the input layer. Instead, ENN relies on a layer named "states" to learn the temporal pattern within the time-series input (e.g., a power sequence measured at fixed intervals). The key difference between states (or a state layer) and an ordinary hidden layer is that there are local feedback connections within the state layer. In other words, a state neuron takes its output at the last moment as a part of input, which consequently makes a single state layer equivalent to a combination of multiple hidden layers in the process of forward propagation.

Elman neural network is essentially a basic form of recurrent neural network with a single hidden layer. ENN is widely adopted in time-series processing because its complexity is adequate for many applications like discrete signal analysis, dynamic systems [34] and predictions [35][36]. Fig. 3 shows the network structure of a typical ENN power model, where $s_q(t)$ denotes the output of a neuron at time t in the state layer. The state layer of ENN contains local feedback (current states take as input the output of previous states) and thus plays the role of memory. This structure enables the hidden layer to retain the impact of previous data input on current feed-forward process and the memory in turn affects the next forward propagation after being updated. Therefore, ENN is an ideal model for processing time series of power consumption.

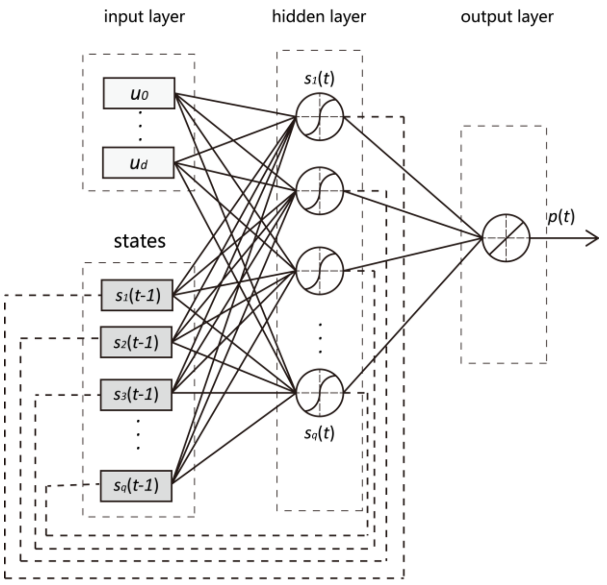


Fig. 3. The network structure of ENN power model.

Elman Neural Network is usually trained using Back Propagation Through Time (BPTT) algorithm which shares similar backward propagation process with BP except that error is propagated back through time in the state layer. A specific hyper-parameter, *steps_back*, is needed to limit the back-propagation distance during ENN's training, and its optimal value basically depends on the temporal pattern of the target sequence.

According to BPTT training process, error in state layer propagates through time because the output of a state unit at time t depends on its output at time t-1:

$$s_t = f(Wx_t + Us_{t-1} + b) \tag{1}$$

where f is the activation function, s_t and x_t denote the vectors of state layer output and features input, respectively. W and U are the matrices that consist of vectors of weights corresponding to performance features and local feedback of states, respectively. b is the bias vector. Let y_t and E_t denote ENN model's output and the loss computed at time t, respectively. Thus, the gradient of loss function with respect to W and U, according to the chain rule, can be respectively formulated as bellows:

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial s_k}{\partial W}\left(\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial y_t}{\partial s_t}\frac{\partial E_t}{\partial y_t} \tag{2}$$

$$\frac{\partial E_t}{\partial U} = \sum_{k=0}^{t} \frac{\partial s_k}{\partial U}\left(\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial y_t}{\partial s_t}\frac{\partial E_t}{\partial y_t} \tag{3}$$

where $\partial s_j/\partial s_{j-1}$ is the derivative of the activation function. Examining (1) (2) and (3), it can observe that the term $\prod_{j=k+1}^{t} \partial s_j/\partial s_{j-1}$ exponentially increases (i.e., exploding gradient) or approaches zero (i.e., vanishing gradient) if we adopt commonly-used activation functions such as *sigmoid* or *tanh*. This phenomenon leads to the major limitation of ENN model but can be optimized by leveraging different activation functions, imposing limits on gradients, applying truncated BPTT algorithm [32], and using evolutionary methods [33] to accelerate training.

Table 1 summarizes the advantages and limitations of the ANN structures that are applicable to modeling time series of power.

From Table 1 we can see that ENN power model has clear advantages over other neural network structures. ENN model works independently on historical data and is able to learn the association between power consumption data regarding time dimension. Therefore, we propose to use ENN to build cloud server power model, namely PCM-ENN. To reduce the complexity in training, we leveraged BPTT algorithm with a short time step.

4 PCM-ENN

We introduce the proposed power consumption model based on Elman neural network (PCM-ENN) in this section. First, the network structure is introduced including the design of input layer, output layer, state layer, and the selection of activation function. We then discuss the methods we applied to model training optimization.

4.1 Model Design

The proposed PCM-ENN is an end-to-end black box power consumption model. Black box model stands for modeling method that treats the target system as a whole despite of its internal functioning. Multivariate regression model, for instance, is a typical black box model as the coefficients are usually not interpretable, whereas power models at component level [11] are white box models since system power is clearly decomposed as the summation of individual components' power.

We selected CPU utilization, memory usage, disk throughput and disk IO request rate as our model's input, considering that they are the most commonly-used, easy-to-sample features for cloud server power models. The size of state layer is a tunable hyper-parameter in our model and will be decided through experiments. We will discuss it in the experimental evaluation section. We choose *tanh* (i.e., $f(x) = (e^x - e^{-x}) / (e^x + e^{-x})$) and *purelin* (i.e., $f(x) = x$) as the activation functions of state layer and output layer, respectively.

We set our model's input layer size to 4 (d=4), state layer size q (q should be determined through experiment) and only one neuron in the output layer. The output of PCM-ENN is an estimate of power consumption. We adopt different activation functions, weight initializers and bias initializers for the state layer and output layer. Note that we use *tanh* as the activation function of each state neuron because truncated BPTT can well eliminate the problem of vanishing or exploding gradients.

TABLE 1.
THE ADVANTAGES AND LIMITATIONS OF USING BPNN, NAR, NARX OR ENN TO MODEL POWER CONSUMPTION

Model	Advantages	Limitations
Sliding Window BPNN model	1. Independence on any system performance counters.	1. The model may have poor accuracy as power consumption changes with large uncertainty. 2. It cannot work without historical power data
BPNN model with only features input	1. The model receives the same input as regression model and is able to achieve better accuracy and support incremental training.	1. It is unable to learn the temporal relation among time-series records.
NAR power model	1. Independence on any system performance counters.	1. The model may have poor accuracy as only historical data is utilized.
NARX power model	1. NARX introduces system performance features to its input layer while maintaining the feedback from the output layer. 2. It is able to learn the temporal pattern in the time series input.	1. Difficult to determine input time window size.
ENN power model	1. Independence on historical power data records. 2. ENN model's state layer has local feedback and is able to learn temporal relation between time-series records.	1. High training complexity.

4.2 Model Training Optimization

Elman neural network, as a kind of recurrent neural network, is usually trained using Back Propagation Through Time (BPTT) algorithm. However, gradients diminish rapidly during back-propagation with long input sequence. Thus, we used Truncated BPTT algorithm to accelerate the training process of PCM-ENN. Truncated BPTT imposes a limit on the steps that training error propagates back through time in the state layer. Besides, we made use of regularization and early-stopping to eliminate over-fitting. Regularization is applied by adding an extra term to the calculation of loss for restricting the weight values. Early-stopping is a technique that terminates the training process according to some rule for the conservation of best model parameters before over-fitting occurs. The stopping rule we adopted is that validation error remains non-decreasing for a number of epochs. We did not adopt "dropout" since PCM-ENN is not of high complexity in structure.

5 EXPERIMENTAL EVALUATION

In this section we briefly introduce experimental setup including the power consumption data set and parameter settings. Then we demonstrate the evaluation results of PCM-ENN against some baselines including NARX power model, the multivariate linear regression model, and the power monitoring software Joulemeter released by Microsoft.

5.1 Experimental Setup

We implemented, trained and evaluated PCM-ENN and NARX network using Python language based on the machine learning framework TensorFlow. The libraries we used mainly include Tensorflow[1] 1.6.0, Numpy 1.14.2, Scipy 1.0.1, and Scikit-learn 0.19.1. With Scikit-learn library, we utilized the class *LinearRegression* in the module *linear_model* to implement the multivariate linear regression model. We launched performance counters to collect CPU utilization, memory usage, disk throughput rate, and disk I/O operation rate while system power was obtained via an external metering device (model: Wattsup?Pro) with logging function.

To cover the diversity of server hardware and operating systems, we investigate the accuracy of power models on two data sets, which were sampled on servers with completely different hardware and operation systems running different suites of benchmarks.

On a Tower Server with Windows

The first data set used in our experiment was sampled on a Dell PowerEdge T110 server with Windows Server 2008 R2 sp1 as operating system. We use *PCMark*[2] 7(subversion: v1.4.0) to generate different types of workload (e.g., computation-intensive and storage-intensive).

[1] Tensorflow. https://tensorflow.google.cn/
[2] PCMark 7. https://www.futuremark.com/benchmarks/pcmark7

PCMark contains multiple benchmark suites. To enable the model to generalize to a wide range of workload, different benchmark suites were exploited including Computation Suite (CPU-intensive), System Storage Suite (Memory and I/O-intensive), and Productivity Suite (mixed workload, for validation and testing).

On a Blade Server with Linux

The second data set used was sampled on a Blade server (model: Dell R730) with Centos 7.6 (kernel version: 3.10.0) as operating system. We use a combination of Linux benchmarks to generate diverse workload. Specifically, they are CloudSuite 3.0 [37], Sysbench[3] and IOzone[4]. Sysbench and IOzone are common benchmarks for testing Linux systems performance. CloudSuite, developed by a community, is a benchmark suite for cloud services consisting of eight popular applications in datacenters. They are based on real-world workload and represent practical setups in production environments.

Table 2 describes the two data sets we used in the experiments. Each data set is divided into training set and test set from which a part of data is used for validation. We normalized all the features as well as power in pre-processing since ENN is sensitive to the range of input and output values. We applied a simple min-max transformation where all the fields are mapped to [0, 1]. The transformation is formulated as (4):

$$\tilde{z}_d = \frac{z_d - min(z_d)}{max(z_d) - min(z_d)}, d = 1,2,3,4,5 \tag{4}$$

where $\tilde{z}_d$ is the dth field (totally five fields including four features and power consumption) after standardization and z_d is the original value. Standardization of model's input and output makes the model incompletely end-to-end. Therefore, we adopt a pre-processing module in the overall workflow to standardize input in advance and a post-processing module to inversely convert the model's output to the range of power consumption.

As mentioned in section 4.2 we applied regularization and early-stopping to reduce the risk of over-fitting. After rounds of tests we finally selected L2 regularization and set *lambda* (impact of regularization term) to 0.0003. Observing that the model usually converged within tens of epochs, we used a small value of *patience* in order to optimize the training process.

5.2 Hyper-parameter Optimization

We mainly discuss two hyper-parameters, namely *state_size* and *steps_back*, which make significant impact on the performance of PCM-ENN. We in the experiment set *state_size* to 6, 9, 12 and 15, in turn, while *steps_back* were set to 1, 2 and 3, in turn. The result on data set 1 is shown in Fig. 4. Similar results were observed on data set 2.

It is notable from Fig. 4 that the best setting of *steps_back* is 1. The reason is that the temporal correlation becomes much weaker when the time interval between two records increases. We also investigated the number of training epochs (iterations through all batches) before the model's convergence with different *state_size* and *steps_back*. The result is shown in Table 3, with the first number indicating the number of epochs needed and the second mean relative error (MRE) of the trained model. It can be noted that the model converged quickly when we set a short back-propagation time step (*steps_back* = 1 or 2). The reason is that the truncated BPTT algorithm effectively accelerated the training process by limiting the back-propagation of errors along the dimension of time. We also find that the number of neurons in the state layer (*state_size*) does not make significant impact on the model's accuracy. It implies that the complexity of ENN with dozens of state neurons is sufficient for modeling server power consumption on our data set. According to the result, we finally set *steps_back* to 1 and the number of neurons in state layer to 12.

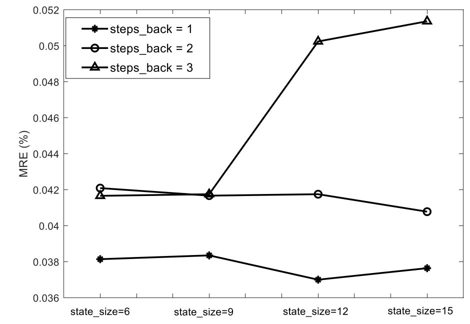


Fig. 4. PCM-ENN's mean relative error with different *state_size* and *steps_back* settings.

5.3 Experimental Results

This section reports our experimental evaluation of the proposed power consumption model based on ENN. For comparison, we trained an NARX network on the same dataset and tuned its hyper-parameters until attaining its best accuracy. We also fitted multivariate linear regression models using Scikit-learn[5], with the same set of features on the two data sets mentioned in section 5.1, as shown in (5) and (6), respectively:

$$\tilde{p}_{LR,1} = 26.57 + 25.94u_{cpu} + 0.97u_{mem} \\ +0.03m_{disk} - 0.0002r_{disk} \tag{5}$$

$$\tilde{p}_{LR,2} = 41.72 + 27.62u_{cpu} + 13.14u_{mem} \\ -0.023m_{disk} - 0.0054r_{disk} \tag{6}$$

where u_{cpu}, u_{mem}, m_{disk} and r_{disk} are CPU utilization, memory usage, disk throughput rate and IO request rate, respectively. We choose linear regression and NARX network as baselines because they are the most representative, commonly used models for regression problems that are time-independent or time-relevant, respectively. In addition, we launched a Windows power monitoring software, named Joulemeter [16], to estimate server power on a regular basis, and we set its interval to 1

[3] Sysbench. https://github.com/akopytov/sysbench
[4] IOZone. http://www.iozone.org/
[5] https://scikit-learn.org/stable/

second.

TABLE 2.
DESCRIPTIONS OF THE POWER DATA SETS USED IN OUR EXPERIMENTS

	Data set 1 (T110 tower, windows)					**Data set 2** (R730 blade, Linux)				
	#samples	min	max	avg.	std. dev.	#samples	min	max	avg.	std. dev.
CPU utilization (%)	1850	0.00	0.69	0.16	0.11	3568	0.00	0.99	0.19	0.21
Memory usage (GB)	1850	1.75	2.54	1.98	0.22	3568	4.66	7.41	5.13	0.40
Disk throughput (MB/s)	1850	0.00	133.56	12.94	21.14	3568	0.00	148.67	57.72	50.12
Disk operation rate	1850	0.00	862.86	92.05	120.82	3568	0.00	881.00	234.47	202.18
Power Consumption (Watts)	1850	27.40	49.70	32.74	3.14	3568	106.20	154.30	114.09	7.11

TABLE 3.
THE NUMBER OF TRAINING EPOCHS AND MODEL'S MRE WITH DIFFERENT SETTINGS OF STATE_SIZE AND STEPS_BACK

	steps_back = 1	**steps_back** = 2	**steps_back** = 3
state_size = 6	7, 0.03814	7, 0.04209	8, 0.04166
state_size = 9	7, 0.03835	7, 0.04167	21, 0.04175
state_size = 12	**7, 0.03700**	7, 0.04175	8, 0.05024
state_size = 15	7, 0.03764	8, 0.04078	8, 0.05136

The functioning of Joulemeter relies on several built-in component power models of three major components (CPU, memory and hard disk). Its Developers from Microsoft adopt a linear model for each of them.

The estimated server power consumption by PCM-ENN, NARX model, linear regression model and Joulemeter (exclusive on Windows) on the two test data sets are drawn in Fig. 5 and Fig. 6, respectively. Actual power (ground truth) measured by the external meter is shown as the black solid line.

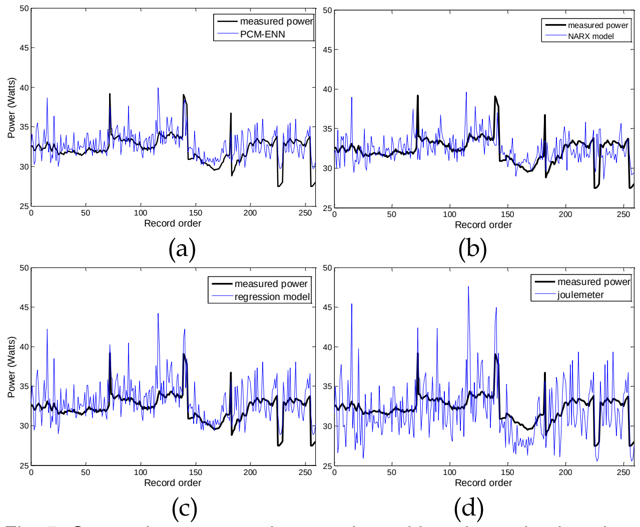


Fig. 5. Comparing measured power data with estimated values by (a) PCM-ENN, (b) NARX network model, (c) linear regression model, and (d) Joulemeter over the running of windows benchmark (PCMark 7) on Dell PowerEdge T110 tower

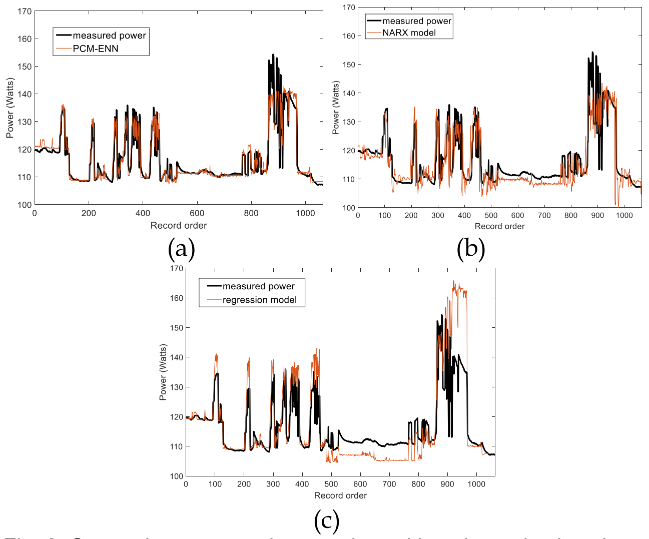


Fig. 6. Comparing measured power data with estimated values by (a) PCM-ENN, (b) NARX network model, and (c) linear regression model over the running of Linux benchmarks (CloudSuite + Sysbench + IOzone) on Dell PowerEdge R730 Blade

PCM-ENN precisely estimated the server's power consumption throughout the test period though some of the power peaks/troughs were under/over-estimated (e.g., the trough starting at the 227th second in data set 1 and the peak around the 850th second in data set 2). We think the reasons behind are two-fold. On one hand, the memorized "states" of the hidden layer "smooth" the estimate sequence but, from a holistic prospective, improve overall accuracy. On the other, extreme values are rare and may not appear in the training set to learn. For instance, server power reached 154.3 Watts at the 882nd second on R730 but the maximum value in the training set is merely 138.0, which probably leads to misestimate in extremely power-intensive situations. The estimates by Joulemeter and regression model deviated significantly from the real values in many periods (e.g., period from 130 to 140 on data set 1 and that from 530 to 760 on data set 2) and tended to overestimate the peaks. NARX network yielded better results than regression model, but it under-estimated the server's power consumption at the ending period of both tests.

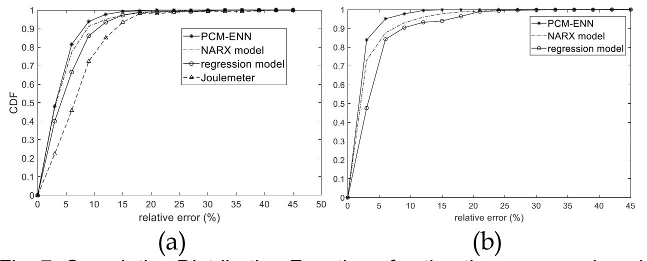


Fig. 7. Cumulative Distribution Function of estimation error produced by PCM-ENN, NARX model, regression model and Joulemeter (exclusive on Windows) throughout the test on (a) Dell PowerEdge T110 Windows server and (b) Dell PowerEdge R730 Linux Server

To further clarify their accuracy, we computed every power model's mean absolute error (MAE), root mean square error (RMSE), maximum relative error (max_RE), and mean relative error (MRE). The statistical results on both data sets are summarized in Table 4.

We can conclude from Table 4 that the proposed power model PCM-ENN is of remarkable accuracy in the tests on the two heterogeneous machines, with relative error smaller than 4% and 2%, respectively. The result also shows its strong ability to generalize learning from the fact that PCM-ENN's maximum errors are apparently smaller than NARX, regression model, and Joulemeter even when the test sets contain more extreme values than the training sets.

We drew CDF (Cumulative Distribution Function) curves to better demonstrate how the four power estimators performed on the two test platforms. The curves are obtained by calculating relative error on each power record. The results are shown in Fig. 7. It can be observed from the error distribution (Fig. 7) that the deviations of PCM-ENN's output from real power are basically within a small margin, with 50 percent/80 percent of the estimate values above 95% accurate, while over 90 percent/97 percent of them above 90% accurate on data set 1 and 2, respectively. For regression model and the monitoring software Joulemeter, it is notable that more than 10 percent of their estimated values are below 90% accurate on the tower server. An important reason is that they are too sensitive to the changes of input features and, consequently, tend to make significantly jittering estimates on servers' power, especially in case power peaks and troughs appear. Besides, we observe that only a tiny fraction of estimates by PCM-ENN is of relative error over 12%, whereas the regression model and Joulemeter are more likely to make deviated estimates. NARX network performed slightly better than regression model, with relative error averaging at 4.3% and 3.1% on the two servers respectively.

Fast training is also a key feature especially in heterogeneous environments. We summarize the training cost of PCM-ENN for both T110 and R730 in Table 5. Besides, we also evaluate the cost of re-training when transplanting the model for T110 to fit the power behavior of R730. In this case, the original model saved for T110 can be reckoned as a pre-trained one. From the result we can see that the training finishes in a few seconds (on a personal notebook PC) as only less than 10 epochs are needed for convergence. Table 5 also shows a slightly reduction of time cost (roughly 6.7%) when pre-training is applied.

Overall, the experimental evaluation demonstrates that our proposed model based on Elman neural network is of high accuracy for server power estimation. It also reflects the model's ability to generalize as it can well perform on comprehensive workload on different hardware and platforms whilst supporting fast training and pre-training.

TABLE 4.
STATISTICAL COMPARISON OF POWER MODELS' ACCURACY ON DATA SET 1 (WINDOWS BENCHMARKS, DELL POWEREDGE T110) AND DATA SET 2 (LINUX BENCHMARKS, DELL POWEREDGE R730)

	Data set 1 (T110 tower, windows)				**Data set 2** (R730 blade, Linux)			
Power Estimator	**MAE(Watts)**	**RMSE**	**Max_RE**	**MRE**	**MAE(Watts)**	**RMSE**	**Max_RE**	**MRE**
Regression model	1.905	2.146	0.308	0.050	4.976	7.991	0.398	0.041
Joulemeter	2.345	2.945	0.410	0.072	-	-	-	-
NARX	1.382	1.901	0.247	0.043	3.795	6.049	0.267	0.031
PCM-ENN	**0.976**	**1.514**	**0.181**	**0.037**	**2.032**	**3.538**	**0.229**	**0.016**

TABLE 5.
TRAINING OVERHEADS OF PCM-ENN FOR SERVERS T110 AND R730

Target machine	MSE	# of epochs	Training time (s)
T110	0.011	7	4.59
R730	0.003	8	7.74
R730 (pre-trained)	0.003	7	7.22

6 CONCLUSIONS

In this paper we first summarize several forms of neural networks that can be applied to cloud server power modeling on time-series datasets. Through analyzing their advantages and limitations, we figure out that ENN is the most suitable among the candidate models for estimating server power consumption as a time series in heterogeneous environment like clouds. Then with the aim of realizing real-time power consumption estimation we propose a novel cloud server power model, namely PCM-ENN. PCM-ENN is an end-to-end black box model that takes easy-to-collect, platform-independent performance features as input, and outputs an estimated value of server power. We trained the model on two datasets collected from a tower server and a blade server by running multiple types of workload after empirically determining the model's optimal hyper-parameters. The evaluation results of PCM-ENN with other baseline models show its high accuracy, fast training as well as strong ability to generalize what it learned from a limited amount of training data to a more complex test scenario with mixed workload.

We plan to focus our future work on increasing the model's feature dimension by considering more performance and state features. With a larger number of available features, feature selection and dimensionality reduction probably need to be done. Alternatively, principal component methods can be adopted to build separate feature sets for different types of workload, which enables the power estimator to be workload-aware for accuracy improvement. We are also trying to further improve our power model's accuracy and efficacy through exploring more complex forms of networks like streaming models.

ACKNOWLEDGEMENT

This work is supported by National Natural Science Foundation of China (Grant Nos. 61772205, 61872084), Guangdong Science and Technology Department (Grant Nos. 2017B010126002, 2017B090901061), Guangzhou Science and Technology Program key projects (Grant No. 201802010010, 201807010052, 201907010001, 201902010040), Nansha Science and Technology Projects (Grant No. 2017GJ001) and the Fundamental Research Funds for the Central Universities, SCUT.

REFERENCES

[1] P. Delforge. (2014, Feb. 06). America's data centers consuming and wasting growing amounts of energy. [Online]. Available: https://www.nrdc.org/resources/americas-data-centers-consuming-and-wasting-growing-amounts-energy

[2] Emerson Network Power. (2009, May 28). Survey of Emerson Network Power's Data Center Users Group Shows Energy Efficiency and Monitoring on the Rise [Online]. Available: https://news.thomasnet.com/companystory/survey-of-emerson-network-power-s-data-center-users-group-shows-energy-efficiency-and-monitoring-on-the-rise-827187

[3] W. Lin and W. Wu, "Energy consumption measurement and management in cloud computing environment," *Ruan Jian Xue Bao/Journal of Software*, vol. 27, no. 4, pp. 1026-1041, 2016 (in Chinese). Doi: 10.13328/j.cnki.jos.005022.

[4] J.C. Mccullough and Y. Agarwal, J. Chandrashekar, S Kuppuswamy, A.C. Snoeren and R.K. Gupta, "Evaluating the effectiveness of model-based power characterization," in: *Usenix Annual Technical Conference*, CA, 2011.

[5] W. Zhu, Y. Zhuang, and L. Zhang, "A three-dimensional virtual resource scheduling method for energy saving in cloud computing," *Future Generation Computer Systems*, vol. 69, 66-74, 2017.

[6] X. K. Li, C. H. Gu, Z. P. Yang, and Y. H. Chang, "Virtual machine placement strategy based on discrete firefly algorithm in cloud environments," *International Computer Conference on Wavelet Active Media Technology and Information Processing*, IEEE, 2016, pp. 61-66.

[7] W. Lin, W. Wu, and J. Z. Wang, "A heuristic task scheduling algorithm for heterogeneous virtual clusters," *Scientific Programming*, vol. 2016, no. 5, pp.1-10, 2016. Doi: 10.1155/2016/7040276

[8] Z. Niu, B. He, and Liu, F, "Not All Joules are Equal: Towards Energy-Efficient and Green-Aware Data Processing Frameworks," *IEEE International Conference on Cloud Engineering*, 2016, pp. 2-11, IEEE.

[9] W. Wu, W. Lin, and Z. Peng, "An intelligent power consumption model for virtual machines under CPU-intensive workload in cloud environment," *Soft Computing*, vol. 21, no. 19, pp. 5755-5764, 2017.

[10] C. H. Hsu and S. W. Poole, "Power signature analysis of the SPECpower_ssj2008 benchmark," in: *IEEE International Symposium on PERFORMANCE Analysis of Systems and Software*, 2011, pp. 227-236, IEEE.

[11] W. Lin, W. Wu, H. Wang, J. Z. Wang, and C. H. Hsu, "Experimental and quantitative analysis of server power model for cloud data centers," *Future Generation Computer Systems*, vol. 86, pp. 940-950, 2016

[12] J. Kumar and A.K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Generation Computer Systems*, vol. 81, pp. 41-52, 2017.

[13] Y. Chang, R. Chang, and F. Chuang, "A Predictive Method for Workload Forecasting in the Cloud Environment," *Lecture Notes in Electrical Engineering*, vol. 260, pp. 577-585, 2014.

[14] S. Gupta, V. Singh, A.P. Mittal, and A. Rani, "Weekly Load Prediction Using Wavelet Neural Network Approach," in: *Second International Conference on Computational Intelligence & Communication Technology*, IEEE Computer Society, pp. 174-179, 2016.

[15] IBM. (2014, June 12). IBM Systems Director. [Online] Available: https://www.ibm.com/support/knowledgecenter/en/POWER6/iphb1/iphb1directoragents.htm

[16] Microsoft. (2010, Feb. 23). Joulemeter: Computational Energy Measurement and Optimization. [Online] Available: https://www.microsoft.com/en-us/research/project/joulemeter-computational-energy-measurement-and-optimization

[17] CLOUDS Laboratory. (2016, May 23). CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. [Online] Available: http://www.cloudbus.org/cloudsim/

[18] M. Dayarathna, Y. Wen, and R. Fan, "Data Center Energy Consumption Modeling: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732-794, 2017.

[19] W. Lin, H. Wang, and W. Wu, "A power monitoring system based on a multi-component power model," *International Journal of Grid & High Performance Computing*, vol. 10, no. 1, pp. 16-30, 2018.

[20] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "RVLBPNN: A workload forecasting model for smart cloud computing," *Scientific Programming*, no. 2016, pp.1-9, 2016.

[21] J. Kumar, R. Goomer, and A. K. Singh, "Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model for Cloud Datacenters," *Procedia Computer Science*, vol. 125, pp. 676-682, 2018.

[22] Z. Chen, Y. Zhu, Y. Di, and S. Feng, "Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network," *Comput Intell Neurosci*, no. 2015, pp. 1-14, 2015

[23] J.J. Prevost, K.M. Nagothu, B. Kelley, and J. Mo, "Prediction of cloud data center networks loads using stochastic and neural models, in: *International Conference on System of Systems Engineering (SoSE)*," IEEE Computer Society. 2011, pp. 276-281.

[24] N. Roy, A. Dubey, and A. Gokhale, "Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting," in: *IEEE International Conference on Cloud Computing*, IEEE, 2011, pp. 500-507.

[25] L.G.B. Ruiz, R. Rueda, M.P. Cuéllar, and M.C. Pegalajar, "Energy consumption forecasting based on Elman neural networks with evolutive optimization," *Expert Systems with Applications*, vol. 92, pp. 380-389, 2017.

[26] Q. Li, B. Guo, Y. Shen, J. Wang, Y. Wu, and Y. Liu, "An Embedded Software Power Model Based on Algorithm Complexity Using Back-Propagation Neural Networks," in: *Green Computing and Communications*, IEEE Computer Society, 2011, pp. 454-459.

[27] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A.A. Bhattacharya, "Virtual machine power metering and provisioning," in: *ACM Symposium on Cloud Computing*, ACM, 2010, pp. 39-50.

[28] G. Tang, W. Jiang, Z. Xu, F. Liu, and K. Wu, "Zero-cost, fine-grained power monitoring of datacenters using non-intrusive power disaggregation," In: *16th Annual Middleware Conference*, ACM, 2015, pp. 271-282.

[29] G. Tang, W. Jiang, Z. Xu, F. Liu, and K. Wu, "NIPD: non-intrusive power disaggregation in legacy datacenters," *IEEE Transactions on Computers* 66(2), pp. 312-325, 2017.

[30] P. POPA. (2006, Oct.). Managing server energy consumption using IBM PowerExecutive, IBM Systems and Technology Group Tech. Rep. [Online] Available: http://images.incisivemedia.com/v7_static/pdf/vnu/optit-wp-stg-power-executive.pdf

[31] J.L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14 no. 2, pp. 179–211, 1990. doi:10.1016/0364-0213(90)90002-E.

[32] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, vol. 1, no.4, pp. 339-356, 1988.

[33] X. Cai, N. Zhang, G.K. Venayagamoorthy, and D.C. Wunsch II, "Time series prediction with recurrent neural networks trained by a hybrid PSO–EA algorithm," *Neurocomputing*, vol. 70, no. 13-15, pp. 2342-2353, 2007.

[34] X.Z. Gao, X.M. Gao, and S. J. Ovaska, "A modified Elman neural network model with application to dynamical systems identification," In *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, 1996, October, vol. 2, pp. 1376-1381, IEEE.

[35] J.J. Wang, W. Zhang, Y. Li, J.Z. Wang, and Z. Dang, "Forecasting wind speed using empirical mode decomposition and Elman neural network," *Applied Soft Computing*, vol. 23, no. 452-459, 2014.

[36] T. Koskela, M. Lehtokangas, J. Saarinen, and K. Kaski, "Time series prediction with multilayer perceptron, FIR and Elman neural networks," In: *World Congress on Neural Networks*, 1996 September, pp. 491-496, INNS Press San Diego, USA.

[37] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, et al, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," In *ACM SIGPLAN Notices*, 2012, March, vol. 47, no. 4, pp. 37-48. ACM.

[38] K. Shen, A. Shriraman, S. Dwarkadas, X. Zhang, and Z. Chen, "Power containers: an OS facility for fine-grained power and energy management on multicore servers," In *ACM SIGPLAN Notices*, 2013 March, vol. 48, no. 4, pp. 65-76, ACM.

[39] K. Li," Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management," *IEEE Transactions on Cloud Computing*, vol .4, no. 2, pp. 122-137, 2016.

[40] U. Wajid, C. Cappiello, P. Plebani, B. Pernici, N. Mehandjiev, M. Vitali, M. Gienger, K. Kavoussanakis, D. Margery, D.G. Perez, and P. Sampaio, "On achieving energy efficiency and reducing CO 2 footprint in cloud computing," *IEEE transactions on cloud computing*, vol .4, no. 2, pp. 138-151, 2016.

[41] M. Nir, A. Matrawy, and M. St-Hilaire, "Economic and energy considerations for resource augmentation in mobile cloud computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 99-113, 2018.

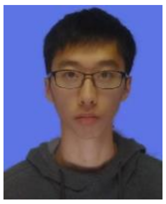
[42] W. Jiang, F. Liu, G. Tang, K. Wu, and H. Jin, "Virtual machine power accounting with shapley value," In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017 June, pp. 1683-1693, IEEE.

[43] W. Jiang, S. Ren, F. Liu, and H. Jin, "Non-IT energy accounting in virtualized datacenter," In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2018 July. pp. 300-310. IEEE.

Wentai Wu received the Bachelor and Master degrees in computer science from South China University of Technology in 2015 and 2018, respectively. Currently, he is a Ph.D. candidate in Computer Science supervised by Prof. Ligang He with the Department of Computer Science, the University of Warwick, United Kingdom. His research interests mainly include parallel and distributed computing, cloud computing, energy-efficient computing and time series processing.

Weiwei Lin received his B.S. and M.S. degrees from Nanchang University in 2001 and 2004, respectively, and the PhD degree in Computer Application from South China University of Technology in 2007. Currently, he is a professor in the School of Computer Science and Engineering, South China University of Technology. His research interests include distributed systems, cloud computing, big data computing and AI application technologies. He has published more than 80 papers in refereed journals and conference proceedings. He is a senior member of CCF.

Ligang He received the Ph.D. degree in Computer Science at the University of Warwick, United Kingdom, and worked as a post-doctoral researcher at the University of Cambridge, UK. From 2006, he worked in the Department of Computer Science at the University of Warwick as Assistant Professor and then Associate Professor. His research interests focus on parallel and distributed processing, Cluster, Grid and Cloud computing. He has published more than 100 papers in international conferences and journals, such as IEEE Transactions on Parallel and Distributed Systems, IPDPS, CCGrid, MASCOTS. He has been a co-chair or a member of the program committee for a number of international conferences, and been the reviewers for many international journals, including IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, etc. He is a member of the IEEE.



Guangxin Wu received the B.E. degree in computer science from the South China University of Technology in 2018. He is currently a master student in Computer Science at South China University of Technology. His research interests include search engine and cloud computing.

Ching-Hsien Hsu is a professor in the Department of Computer Science and Information Engineering, Chung Hua University, Taiwan, and a distinguished chair professor in the School of Computer and Communication Engineering, Tianjin University of Technology, China. His research includes high-performance computing, cloud computing, parallel and distributed systems, big data analytics, ubiquitous/pervasive computing, and intelligence. He has published 200 papers in refereed journals, conference proceedings, and book chapters in these areas. He is the editor-in-chief of the International Journal of Grid and High Performance Computing and International journal of Big Data Intelligence. He is serving as an editorial board member for a number of prestigious journals, including IEEE Transactions on Service Computing, IEEE Transactions on Cloud Computing, etc. He has been acting as an author/co-author or an editor/coeditor of 10 books from Springer, IGI Global, World Scientific, and McGraw-Hill. He has also edited a number of special issues at top journals, such as IEEE Transactions on Cloud computing, IEEE Transactions on Services Computing, IEEE System Journal, Future Generation Computer Systems, Journal of Supercomputing, etc. He received eight times distinguished award for excellence in research and annual outstanding research award through 2005 to 2015 from Chung Hua University. He has been serving as executive committee of Taiwan Association of Cloud Computing from 2008 to 2012; executive committee of the IEEE Technical Committee of Scalable Computing (2008-2012); IEEE Cloud Computing (since 2012). He is a senior member of the IEEE.