Developing Cloud Computing Time Slot-availability Predictions Using an Artificial Neural Network

저자 (Authors)	Alanazi Rayan, Muhammad Ashfaq khan, Fawaz Alhazemi, Hamoud Alshammari, Yunmook Nah
출처 (Source)	IEIE Transactions on Smart Processing & Computing 9(1), 2020.2, 49-57 (9 pages)
발행처 (Publisher)	대한전자공학회 The Institute of Electronics and Information Engineers
URL	http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE09307299
APA Style	Alanazi Rayan, Muhammad Ashfaq khan, Fawaz Alhazemi, Hamoud Alshammari, Yunmook Nah (2020). Developing Cloud Computing Time Slot-availability Predictions Using an Artificial Neural Network. IEIE Transactions on Smart Processing & Computing, 9(1), 49-57.
이용정보 (Accessed)	이화여자대학교 203.255.***.68 2020/05/18 03:52 (KST)

저작권 안내

DBpia에서 제공되는 모든 저작물의 저작권은 원저작자에게 있으며, 누리미디어는 각 저작물의 내용을 보증하거나 책임을 지지 않습니다. 그리고 DBpia에서 제공되는 저작물은 DBpia와 구독계약을 체결한 기관소속 이용자 혹은 해당 저작물의 개별 구매자가 비영리적으로만 이용할 수 있습니다. 그러므로 이에 위반하여 DBpia에서 제공되는 저작물을 복제, 전송 등의 방법으로 무단 이용하는 경우 관련 법령에 따라 민, 형사상의 책임을 질 수 있습니다.

Copyright Information

Copyright of all literary works provided by DBpia belongs to the copyright holder(s)and Nurimedia does not guarantee contents of the literary work or assume responsibility for the same. In addition, the literary works provided by DBpia may only be used by the users affiliated to the institutions which executed a subscription agreement with DBpia or the individual purchasers of the literary work(s)for non-commercial purposes. Therefore, any person who illegally uses the literary works provided by DBpia by means of reproduction or transmission shall assume civil and criminal responsibility according to applicable laws and regulations.

IEIE Transactions on Smart Processing and Computing, vol. 9, no. 1, February 2020
https://doi.org/10.5573/IEIESPC.2020.9.1.049

Developing Cloud Computing Time Slot-availability Predictions Using an Artificial Neural Network

Alanazi Rayan[1]*, Muhammad Ashfaq khan[2], Fawaz Alhazemi[3], Hamoud Alshammari[4], and Yunmook Nah[5]

[1] Department of Computer Science and Engineering, jouf University / Al-Qurayyat, Saudi Arabia rmalanazi@ju.edu.sa
[2] Department of Computer Engineering, Dongguk University / Seoul -30-pildong-ro-1-gil-jung-gu, Korea
{ashfaq_jiskani@yahoo.com}, {ashfaq_jiskani@dongguk.edu}
[3] Department of Computer and Network Engineering, College of Computer Science and Engineering, University of Jeddah / Jeddah, Saudi Arabia fmalhazemi@uj.edu.sa
[4] jouf University / Al-Qurayyat, Kingdom of Saudi Arabia (KSA) hhalshammari@ju.edu.sa
[5] Department of Applied Computer Engineering, Dankook University / Yongin-si, Gyeonggi-do, Korea
{ymnah@dankook.ac.kr}

* Corresponding Author: Alanazi Rayan, rayanmuwafiq@gmail.com

Received August 15, 2019; Revised September12, 2019; Accepted November 27, 2019; Published February 28, 2020

** Regular Paper*

***Abstract*:** Over the last decade, cloud computing has exponentially transformed the ways of computing. In spite of its various advantages, cloud computing suffers from several challenges that affect performance. Two of the fundamental challenges are power consumption and dynamic resource scaling. An efficient resource allocation strategy could help cloud computing to improve overall performance and operational costs. In this paper, we design a novel approach to available time slot prediction in a data node, based on an artificial neural network (ANN), which predicts the time at which the required resources will be available. We conducted experiments on several nodes, obtaining up to 98%, and outperforming state-of-the-art available time slot prediction approaches. We claim that available time–slot prediction for cloud computing based on an ANN will lead to optimum resource allocation and to minimizing energy consumed while maintaining the essential performance level.

***Keywords*:** Artificial neural network, Resource management, Cloud computing, Data center

1. Introduction

Cloud computing with its wide range of applications, has grown exceptionally over the last few years. Organizations can more quickly have their applications fully operational via the cloud. Moreover, clients can store and process their data through the cloud in third-party data centers. Furthermore, the resources of virtual computing as a service over the Internet are available, which has attained remarkable attention from both educational institutes and business organizations around the world. In the traditional computing approach, a user can only have a fixed amount of computing resources, whereas in cloud computing, a client can access as many resources as required [1]. Hence, cloud computing has become a handy platform for companies to reduce their infrastructure costs. Characteristics like on-demand service, scalability, and robustness (among others) have prompted a large number of companies to switch to the cloud. To provide on-demand services to the clients, vast data centers are deployed by service providers. In a cloud computing environment, these data centers work as a backbone. Virtualization is the key concept behind these data centers, which helps in resource sharing between multiple clients via virtual machines (VMs). The data centers require dynamic resource scaling and an efficient allocation policy to maintain quality of service (QoS) along with maximizing gain for the service providers. In the lifetime of a hosted application, the flexibility of adding or removing virtual hardware resources at any time has raised the possibility of advance resource management along with dynamic scaling [1]. For a data center, dynamic resource scaling has become a point of concern for working in an immaculate manner. Resource scaling relies on different aspects—for instance, the current state of the system, upcoming factors, the number of active users, and several other factors. The migration of virtual machines and non-instantaneous initialization makes reactive approaches

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)

ineffective whereas proactive approaches enable resource scaling before the actual demand occurs. However, these methods require information about upcoming workloads in advance. The demand for future resources can be estimated by the identification of the current state of the data center and the historical usage patterns. As the predicted load on a data center identifies the amount of resources to scale up, a reliable and effective prediction system must be set up for accurate estimation of future workloads. This information can be utilized to boost the utilization of resources and consumption of power. Hence, the cloud data center can be set up at low cost and reduces service level agreement violations. The high non-linearity in workloads, and interactions with varying numbers of users are main challenges for precise prediction. Several approaches are available to predict workloads. The average or maximum workload can be predicted for specific time intervals. However, mean, maximum, and other statistical approaches are too fundamental and ineffective for accurate predictions. For example, if the maximum workload prediction model is used, then most of the time, resources will remain unused, whereas if we use a prediction model based on the average workload, then the system will face problems due to a lack of resources. So performance will degrade when there is an increase in workload. The prediction models that utilize these approaches can be deemed poor because they rarely make the right prediction [2]. On the other hand, machine learning (ML) techniques are used to make more accurate prediction models. These approaches utilize historical data as a training window for the prediction of workload throughout a prediction interval [3]. The time between each prediction is defined as the prediction interval.

Therefore, in this paper, we have designed an artificial neural network (ANN) for available time slot prediction of a data node, which predicts the time at which the required resources will be available. Some essential points include the following.

- Conventionally available time slot prediction approaches are not globally optimal, so from the performance point of view more efficient and robust solutions are needed for available time slot prediction in cloud environments.
- Machine learning techniques can be utilized for time slot prediction problems in cloud environments where a large amount of data can be collected and utilized to train the machine learning algorithms that produce very robust and effective results for several time slot prediction problems in a cloud environment.
- We propose an ANN-based state-of-the-art machine learning model for time slot prediction and compared it with conventional approaches. Simulation results show that our ML-based approach is better, compared to traditional techniques, and demonstrate the efficacy of the proposed approach.

The rest of the paper is structured as follows. Section 2 describes the literature in detail regarding time slot prediction using several conventional approaches. Section 3 presents state-of-the-art machine learning approaches based on an ANN for time slot prediction. Section 4 illustrates and discusses the experimental work with the proposed approach. Finally, Section 5 summarizes and concludes the work by providing some possible future research directions.

2. Literature Review

In recent years, lots of research has been done on workload prediction in a data center [4]. In [2], Roy et al. presented a forecasting framework that works on an auto-regressive integrated moving average (ARIMA). In the cloud environment, workload prediction is an active field for research, and researchers have presented different techniques to address the problem. History-based prediction and homeostatic prediction are well-known methods. First, in homeostatic prediction, the prediction of upcoming workloads is based on calculations including the mean of the previous workloads subtracted or added from the current workload [5]. These calculations with subtraction or addition of values can be dynamic or static. In the dynamic approach, estimated values from previous workload instances will be considered for calculation purposes. In a static approach, a fixed number is chosen to perform calculations for workload prediction. However, these history-based methods are the most commonly used, simple prediction models. Future demand prediction through these models is based on pattern extraction by analyzing the previous workloads. Tendency information on previous workload instances is valuable data with which to predict future demand in history-based methods. While the mean of the previous workloads is the backbone in homeostatic models [6], these historical information models are used in data centers to present the workloads exhibiting seasonal trends in the form of a time series [7]. Time series prediction is used extensively in classical methods to predict future demand. Time series data used in time series prediction consist of a set of data points in uniform time intervals at successive points in time–space. Exponential smoothing (ES), ARIMA, along with hidden Markov (HM), moving average (MA), and autoregression (AR) methods are based on time series prediction. An auto regression–based model to predict webserver workload was proposed in [8]. A linear combination of previously observed values is used to estimate the values of future demand prediction in various time instances. In [9], the authors developed a prediction model based on moving averages. Workload prediction using distributed nonlinear optimization techniques was incorporated in the proposed distributed solution using moving averages. In [10], the authors worked on seasonal time series prediction using exponential smoothing models. The proposed model is based on two approaches: the seasonal additive model and the multiplicative seasonal model. These approaches were used in exponential smoothing models to estimate workload. The proposed approach is only suitable for a time series exhibiting seasonal behavior. In [11], the authors analyzed live sports event broadcast service workloads using regression techniques relying on simple statistical models using data from a commercial Internet service provider. In [12], the authors worked on multiple time series approaches to present a workload prediction

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)

model. The working of the model is based on a grouping of similar models for prediction accuracy. The hidden Markov model (HMM) is used in obtained clusters of VMs to distinguish the temporal correlations and variations in workload patterns. In [13], the authors used the Monte Carlo method for workload forecasting. Moreover, machine learning has also been used in workload time series prediction. This machine learning–based prediction is based on provided data to generate a probabilistic score. In [14], the authors presented a novel approach using clustering techniques to predict future data volume. In this framework, the self-organizing map (SOM) model is used for clustering of data, along with support vector machines (SVMs) for prediction purposes. However, the novel framework depends on a threshold value, which is used to decide the number of data points.

The authors in [15], proposed a two-tier architecture for financial time series prediction using k nearest neighbors (kNN). However, kNNs incur a high computational cost to learn from data and to predict future trends. Moreover, a novel method that employs a neural network was presented for model variations in multimedia designs [16]. Those authors proposed a novel model with a combination of linear regression and a resource scaling method based on workload prediction. In [17], the authors proposed a model using a Kalman smoother, statistical learning theory, and support vector regression (SVR). In their proposed model, the authors achieved high prediction accuracy and outperformed the backpropagation network, autoregression, and the standard SVR. An ensemble-based model consisting of five different prediction models was implemented by the authors in [18]. The weights are assigned to each model, and the optimization of weights is based on generic algorithms. The contribution of each model is based on the weights assigned to it. In [19], the authors presented the concept of the sliding window–based linear regression and neural network model. In this method, prediction model evaluation takes place using CPU demand traces. A seasonal ARIMA model was proposed by the authors in [20] to predict upcoming workloads for up to 168 hours. However, the optimization of the model is based on an expert. The authors in [21] used a backpropagation neural network to predict the upcoming workload. The accuracy of predictions was achieved using a stochastic model. In [22], the authors used a pattern-matching algorithm, including Knuth–Morris–Pratt (KMP) string matching to identify similar patterns. However, this algorithm is not effective when the history of the workloads increases. In [23], the authors used a fuzzy neural network, and each resource is presented as a period of flatness or as a period of fluctuation. The system is composed of two prediction layers, which increases the time delay, and makes the workload prediction even more complex. In [24], the authors proposed a resource scaling policy. The main purpose of this resource scaling policy is to monitor the VM CPU usage. In [25], the authors proposed a novel model using a steepest descent learning algorithm and a neural network. The accuracy of the system improved over time by choosing different learning rates. The authors in [26] presented a novel approach that employs a backpropagation learning algorithm. In this

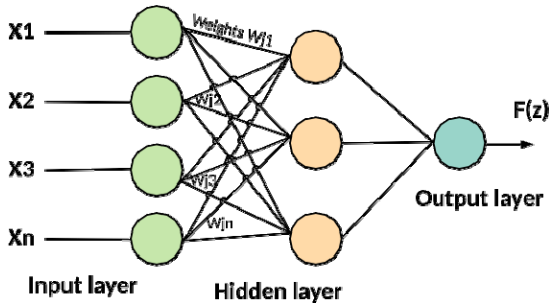


Fig. 1. Artificial Neural Network.

model, the weights of the model for a cloud environment are adjusted according to error trends. There is a strong relationship between the accuracy of the model and the latency levels.

3. Artificial Neural Network for Available Time Slot Prediction

This section explains how an ANN is used to compute the usage of the node. The proposed ANN is based on the formulation of the named node and job manager node information. The artificial neural network shown in Fig. 1 consists of three main components: (1) an input layer, (2) a set of hidden layers, and (3) an output layer. Neurons in each layer vary; the number of neurons in the input layers are normally equal to the number of inputs, and the number of neurons in an output layer is equal to the number of outputs, whereas the number of neurons in hidden layers varies since they need to fit a nonlinear function. Each layer uses an activation function to activate neurons in the forward layer. Each layer consists of a set of inputs and a set of weights. Inputs of a layer, except for the input layer, are the output of the previous layer and the weights that are set using different mechanisms. In the beginning, weights are assigned randomly, and then, they are tuned using a gradient descent algorithm, one of the most widely used algorithms. Consider a set of input variables, and compute the weights for the input variables to obtain the output variables. In summary, ANNs can be represented as a set of inputs, an activation function, and an output. Links between the inputs and a hidden layer are called weights. Weights define the connection strengths between neurons, and are the output value. In the following, we represent the mathematical version of the single-layer neural network.

In the hidden layer, each neuron receives weighted input and bias from other neurons in the input layer, as follows:

$$Zi = (\Sigma Nj-1xj-1w\ k,i - bk) \quad (1)$$

where wk,i is the weight value of the connection between node K and all the nodes in the input layer; xj−1 is the input from the K-th node in the j-th layer, Nj−1 is the total number of nodes in layer j-1, and bk is the bias of the node.

The summation of these weighted values with the activation function is forwarded to compute the output of

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)

the node. The output is computed as follows:

$$yi = f(Zi) \quad (2)$$

The linear function can be used for the activation function and is formulated as follows:

$$fZi = azi + c \quad (3)$$

This equation serves to model the relevant nonlinear behaviors.

An artificial neural network consists of one, two, or three hidden layers with 10 neurons in the input layers and 16, 32, 48, 64, and 128 neurons for each hidden neuron. We have used different hidden layers and a different number of neurons in each hidden layer to determine the impact of hidden layers and the number of neurons in the process of information perception. This method helped us in choosing a better architecture.

To find the best node to perform the job, we first identify the difference between free resources in the running nodes, and we then pass that information to the ANN to compute the probability of whether the node will be suitable for performing the task.

The probability for each node is calculated, and the node with the highest probability is selected to perform the task. The distance between a running node and the resources required has been identified using the Euclidean distance method. The output of the Euclidean distance could be 0 or any positive value. If the output of Euclidean distance is 0, it shows that the node does not have enough resources to perform the task. A minimum value close to 0 would mean that this node has enough resources to perform the task. A higher value would mean that this node is more suitable, since running this node to execute the task will not waste resources, whereas a smaller value greater than 0 would mean that this node has more free resources than required, which means it could be used to perform another job that requires more resources.

Once we find all appropriate nodes that have the required resources, we use the ANN to identify the time span appropriate to run the task, and we forward the information to the resources manager, which then will update the used resources to provide updated information for upcoming jobs.

3.1 Implementation

We implemented our neural network solution in Python 3.6 using the Keras neural network framework. TensorFlow was used as the Keras back end. Output graphs were generated using matplotlib, and data processed by using panda's python open source library. The amount of memory in a data node can be calculated as follows: memory size = (memory per CPU core * CPU core numbers) + data node process memory + data node task tracker memory + OS memory. For a data node with a process memory of 4-8 GB, task tracker memory of 4-8 GB, OS memory of 4-8 GB, and four CPU cores with 4-8 GB units of memory per core, the total memory of the data node is (at least) equal to 4*4+4+4 = 28 GB. In the next section, we explain the second components of the proposed architecture, a modified ant colony for job scheduling.

Table 1. Experimental parameter values.

Parameter	Value
Input Nodes (n)	10
Hidden Nodes (p)	7
Maximum Iterations (Gmax)	250
Training Data Size	60%
Mutation Strategy Learning Period (lpF)	10
Crossover Rate Learning Period (lpCR)	10

4. Experiment and Results

This section discusses the results for the job allocation using artificial neural networks.

Once the jobs are scheduled using a modified ant colony, as discussed in Section 3, the next task is to allocate resources to the job. The overall goal is to execute more jobs using fewer resources. The resource management unit of the architecture takes care of running resources and makes sure there are no idle nodes in the data center.

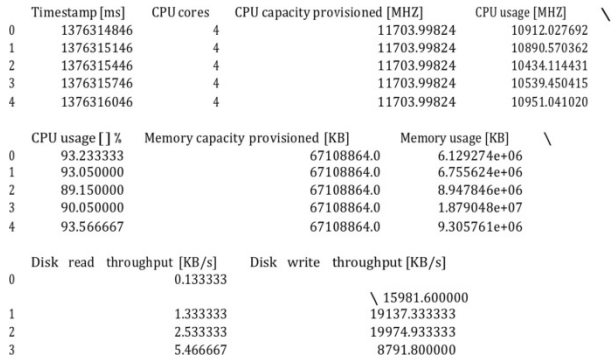
4.1 Dataset

Consider N nodes to be running at a particular time, t, in a data center, and each node is performing allocated tasks. A task, T, has been scheduled first in the queue of pending tasks, and the resources need to be allocated for task T. Information about the required resources comes with the task, and therefore, the task of the artificial neural network is to predict a suitable time (according to the past data) in which the required resources will be available to task T. The dataset used to design and develop the neural network contained resource utilization during one month. The total number of observed instances was 123 million for 1250 machines. Every observation was made after roughly one second. The dataset provides information about the CPUs and their utilization, memory and its utilization, and network bandwidth and its utilization at a particular time stamp. Free resources are calculated by subtracting total resources and utilized resources.

4.2 Construction of the Artificial Neural Network

Once the dataset is understood and examined, the next step is to design and develop an efficient neural network to predict the time at which particular resources will be available so that the task can be scheduled at that particular time. The goal was to design a neural network that can capture maximum patterns in the data without under- or overfitting the data. To achieve this goal, in the first step, a very simple neural network with 10 input neurons, a single hidden layer containing 16 neurons, and the output layer

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)



	Timestamp [ms]	CPU cores	CPU capacity provisioned [MHZ]	CPU usage [MHZ]	\
0	1376314846	4	11703.99824	10912.027692	
1	1376315146	4	11703.99824	10890.570362	
2	1376315446	4	11703.99824	10434.114431	
3	1376315746	4	11703.99824	10539.450415	
4	1376316046	4	11703.99824	10951.041020	

	CPU usage [ ] %	Memory capacity provisioned [KB]	Memory usage [KB]	\
0	93.233333	67108864.0	6.129274e+06	
1	93.050000	67108864.0	6.755624e+06	
2	89.150000	67108864.0	8.947846e+06	
3	90.050000	67108864.0	1.879048e+07	
4	93.566667	67108864.0	9.305761e+06	

	Disk read throughput [KB/s]	Disk write throughput [KB/s]
0	0.133333	\ 15981.600000
1	1.333333	19137.333333
2	2.533333	19974.933333
3	5.466667	8791.800000

Fig. 2. Experimental data sample.

Table 2. Physical elements' descriptions.

Physical elements	Description
Time stamp	number of milliseconds since 1970-01-01
CPU cores	number of virtual CPU cores provisioned
CPU capacity (CPUs requested)	the capacity of the CPUs in terms of MHZ, it equals the number of cores times the speed per core
CPU usage	in MHZ
CPU usage	as a percentage
Memory provisioned (memory requested)	the capacity of the memory of the VM in KB
Memory usage	the memory that is actively used in KB
Disk read throughput	in KB/s
Disk write throughput	in KB/s
Network received throughput	in KB/s
Network transmitted throughput	in KB/s

was created. In the advanced-version neural networks with 10 input neurons, two and three hidden layers with 16 neurons, and one output layer were created. Fig. 3 shows the results for the learning rate of a 16-neuron network. Fig. 3 shows that fewer patterns were captured in a single-layer neural network than with two- and three-hidden–layer neural networks. In the next step, neural networks with single hidden layers containing 32, 48, 64, and 128 neurons were created and tested. The orange lines in Figs. 4-6 show the results for the neural networks with single hidden layers containing 32, 48, and 64 neural networks, respectively. Finally, these neural networks were enhanced to two- and three-hidden–layer neural networks. The observations collected as the result of these experiments show that neural networks with three hidden layers and more neurons are able to capture patterns and produce higher accuracy than neural networks with a single layer or two layers and fewer neurons. Therefore, the experiments helped us conclude that neuron networks with more hidden layers and higher numbers of neurons will capture more patterns than their counterparts. However, another question is when the neural networks will start overfitting. The next section presents the validation results and discusses the

Table 3. Experimentation environment.

Category	Machine / Tools
CPU	2.4 GHz Intel Core i5
GPU	NVIDIA GeForce 1050Ti
RAM	8 GB 1600 MHz DDR3
OS	MacOS Mojave
Language	Python 3.5
Library	Google TensorFlow 1.2 as used in Keras

Table 4. Experimental setup.

Category	Value
Neural model	Artificial Neural Network
Loss function	MSE
Optimizer	RMSProp
Hidden node	16, 32, 48, 64 and 128
Learning Rate	0.001
Mini-batch size	1024

issues of underfitting and overfitting of the neural networks.

(1) Data Generation for Training and Evaluation

We use simulation to generate training and evaluation data, and to generate input and output data for supervised training. Fig. 2 shows the sample dataset. Table 2 shows the physical elements' descriptions.

(2) Experimentation Environment and Setup

Table 3 shows the machine configurations and deep learning frameworks used for our experimentation.

(3) Neural Network Training

Table 4 shows the configuration of the final ANN model optimized from the training data, and other training parameters we selected. The input and output data to train the artificial neural network was generated from the simulated data center in the training process of an artificial neural network, from the number of hidden nodes, from the mini-batch size, and from the total number of executions.

Epochs (the iterations over the entire dataset) and the learning rate are important parameters that affect the training performance. We varied all of these parameters to find the optimized settings for an artificial neural network. Initially, weights and biases were initialized randomly using a standard normal distribution. We use RMSProp as the optimizer and mean squared error (MSE) as the loss function. Through repeated feed-forward propagation of inputs and weights, backpropagation of errors was used to optimize weights as well as the other configuration parameters. Figs. 3-6 show the average localization error during the training in terms of epoch count as we varied the number of hidden layers from one to three, and the number of hidden nodes from 16, 32, 48, and 64, with a batch size of 1024 over 10 iterations of the entire experimental process. As we increased the number of

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)

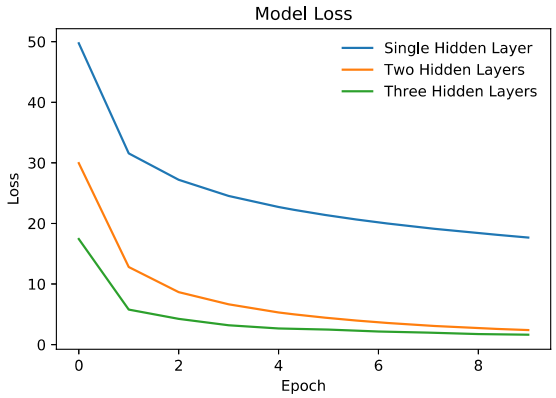


Fig. 3. 16 neurons.

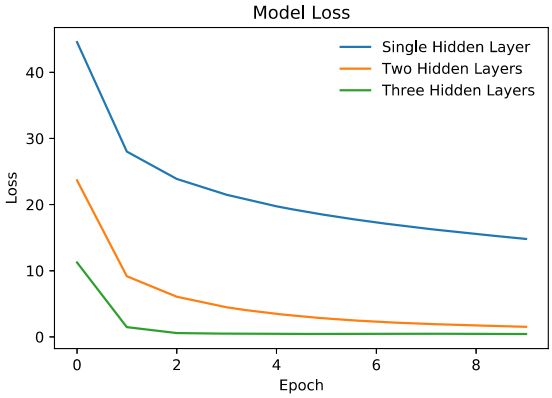


Fig. 4. 32 neurons.

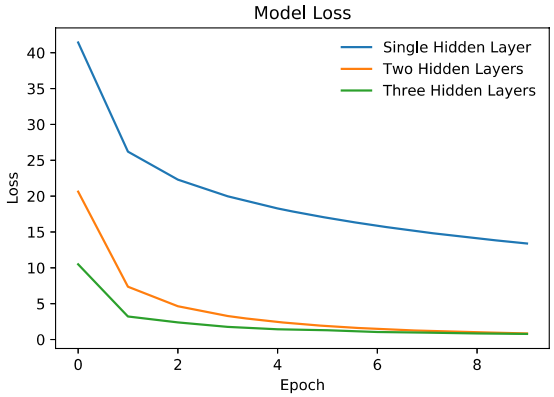


Fig. 5. 48 neurons.

hidden layers and the number of hidden nodes, the training time to reach a target localization error decreased; however, it started overfitting the model. From the experiments, we noticed that error was around zero MSE after fewer iterations with two hidden layers and 32 hidden nodes. Therefore, it can be said that the optimum value can be obtained with two hidden layers and 32 hidden nodes.

4.3 Validation of Artificial Neural Networks

Cross-validation is one of the important components of any machine learning algorithm, which allows users to see the accuracy of the model for alien data. The most popular

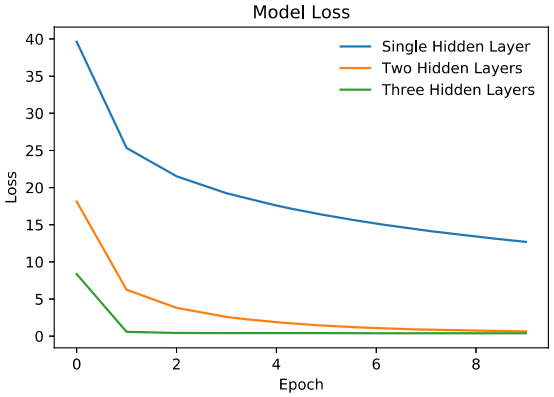


Fig. 6. 64 neurons.

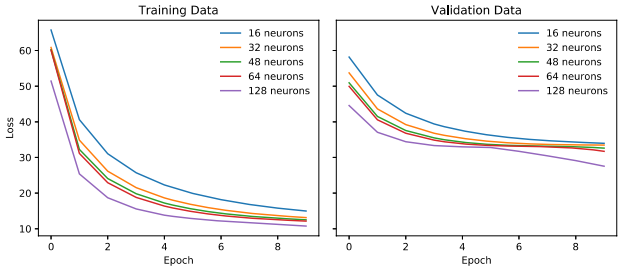


Fig. 7. Single-layer validation.

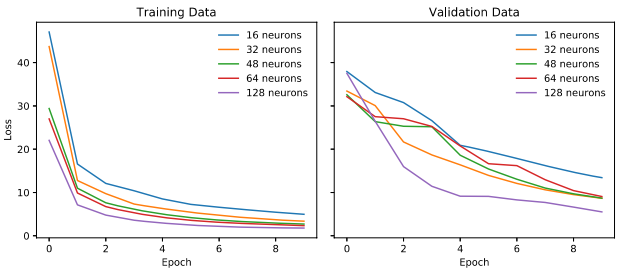


Fig. 8. Two-layer validation.

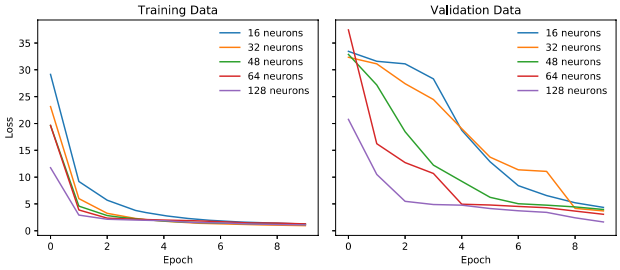


Fig. 9. Three-layer validation.

method, widely used for cross-validation, is training validation split. Therefore, all models designed as the result of this research were also tested using validation data. This dataset was divided into training and validation data using the 80-20 rule (80% of the input data for training, and 20% for testing). Figs. 7-9 present validation results for single-layer, two-layer, and three-layer neural networks, respectively. Results show that a neural network

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)

with a single layer has no overfitting but less accuracy, and this accuracy is reduced for the validation data, which means single-layer neural networks are not able to capture all patterns, so some patterns might still be missing. On the other end, neural networks with three hidden layers produce high accuracy on the training data; however, they result in less accuracy during validation, which means the data overfit a little. However, by comparing results only for validation from Figs. 7-9, we can easily conclude that more layers and more neurons result in improved accuracy, but they need more iterations for training to get them to the convergence point. Conclusively, neural networks can be used to predict the time at which particular resources will be available, based on past data, with high accuracy, and that they can, in the end, result in better management of the resources in the data center. Next, we show an artificial neural network for the data node usage. For prediction results, MSE is commonly used for evaluation of the accuracy of the regression loss function. MSE is the sum of squared distances between the predicted values and the target variable, and can be calculated with the following formula:

$$\text{Root mean squared error RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

5. Conclusion and Future Work

From the last decade, tremendous growth in large-scale data center environments has led the research community to develop effective data center management systems to guarantee productivity with a limited amount of resources. Conventionally, the available time slot prediction approaches are not globally optimal, so from a performance point of view, the most efficient and robust solutions are needed for available time slot predictions in the cloud environment. Machine learning techniques can be utilized for time slot prediction problems in cloud environments where a large amount of data can be collected and utilized to train machine learning algorithms that produce robust and effective results for several time slot prediction problems in cloud environments.

Therefore, in this paper, we designed an artificial neural network for available time slot prediction of a data node, which predicts the time at which the required resources will be available. The results show high accuracy compared to existing approaches.

In the future, this work will be extended by considering various attributes for time slot prediction using several robust machine learning classifiers and state-of-the-art deep learning approaches, such as long short-term memory networks, etc.

Acknowledgement

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2019-2015-0-00363) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation). This research was also supported by MSIT Korea under the National Program for Excellence in SW supervised by the IITP (2017-0-00091).

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation computer systems, vol. 25, no. 6, pp. 599-616, 2009. Article (CrossRef Link)

[2] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in 2011 IEEE 4th International Conference on Cloud Computing, pp. 500-507, IEEE, 2011. Article (CrossRef Link)

[3] K. Cetinski and M. B. Juric, "Ame-wpc: Advanced model for efficient workload prediction in the cloud," Journal of Network and Computer Applications, vol. 55, pp. 191-201, 2015. Article (CrossRef Link)

[4] A. Rayan and Y. Nah, "Resource prediction for big data processing in a cloud data center," IEIE Transactions on Smart Processing & Computing, vol. 7, no. 6, pp. 478-488, 2018. Article (CrossRef Link)

[5] Zhang, Yuanyuan, Wei Sun, and Yasushi Inoguchi. "Predict task running time in grid environments based on CPU load predictions." Future Generation Computer Systems 24.6 (2008): 489-497. Article (CrossRef Link)

[6] S.-R. Kuang, K.-Y. Wu, B.-C. Ke, J.-H. Yeh, and H.-Y. Jheng, "Efficient architecture and hardware implementation of hybrid fuzzy-kalman filter for workload prediction," Integration, the VLSI Journal, vol. 47, no. 4, pp. 408-416, 2014. Article (CrossRef Link)

[7] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen, "An adaptive prediction approach based on workload pattern discrimination in the cloud," Journal of Network and Computer Applications, vol. 80, pp. 35- 44, 2017. Article (CrossRef Link)

[8] T.-H. Li, "A hierarchical framework for modeling and forecasting web server workload," Journal of the American Statistical Association, vol. 100, no. 471, pp. 748-763, 2005.

[9] D. Ardagna, S. Casolari, M. Colajanni, and B. Panicucci, "Dual time- scale distributed capacity allocation and load redirect algorithms for cloud systems," Journal of Parallel and Distributed Computing, vol. 72, no. 6, pp. 796-808, 2012. Article (CrossRef Link)

[10] P. S. Kalekar, "Time series forecasting using holt-winters exponential smoothing," Kanwal Rekhi School of Information Technology, vol. 4329008, no. 13, 2004.

[11] Y. S. Sun, Y.-F. Chen, and M. C. Chen, "A workload analysis of live event broadcast service in cloud," Procedia Computer Science, vol. 19, pp. 1028-1033,

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)

2013. Article (CrossRef Link)
[12] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in 2012 IEEE Network Operations and Management Symposium, pp. 1287-1294, IEEE, 2012. Article (CrossRef Link)
[13] T. Vercauteren, P. Aggarwal, X. Wang, and T.-H. Li, "Hierarchical forecasting of web server workload using sequential monte carlo training," IEEE transactions on signal processing, vol. 55, no. 4, pp. 1286-1297, 2007. Article (CrossRef Link)
[14] L. Cao, "Support vector machines experts for time series forecasting," Neurocomputing, vol. 51, pp. 321-339, 2003. Article (CrossRef Link)
[15] T. Ban, R. Zhang, S. Pang, A. Sarrafzadeh, and D. Inoue, "Referential knn regression for financial time series forecasting," in International Conference on Neural Information Processing, pp. 601-608, Springer, 2013. Article (CrossRef Link)
[16] A. Eddahech, S. Chtourou, and M. Chtourou, "Hierarchical neural networks based prediction and control of dynamic reconfiguration for multilevel embedded systems," Journal of Systems Architecture, vol. 59, no. 1, pp. 48-59, 2013. Article (CrossRef Link)
[17] R. Hu, J. Jiang, G. Liu, and L. Wang, "Efficient resources provisioning based on load forecasting in cloud," The Scientific World Journal, vol. 2014, 2014. Article (CrossRef Link)
[18] V. R. Messias, J. C. Estrella, R. Ehlers, M. J. Santana, R. C. Santana, and S. Reiff-Marganiec, "Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure," Neural Computing and Applications, vol. 27, no. 8, pp. 2383-2406, 2016. Article (CrossRef Link)
[19] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," Future Generation Computer Systems, vol. 28, no. 1, pp. 155-162, 2012. Article (CrossRef Link)
[20] Calheiros, Rodrigo N., et al. "Workload prediction using ARIMA model and its impact on cloud applications' QoS." IEEE Transactions on Cloud Computing 3.4 (2014): 449-458. Article (CrossRef Link)
[21] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in 2011 6th International Conference on System of Systems Engineering, pp. 276-281, IEEE, 2011. Article (CrossRef Link)
[22] E. Caron, F. Desprez, and A. Muresan, "Pattern matching based forecast of non-periodic repetitive behavior for cloud clients," Journal of Grid Computing, vol. 9, no. 1, pp. 49-64, 2011. Article (CrossRef Link)
[23] 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," Computational intelligence and neuroscience, vol. 2015, p. 17, 2015. Article (CrossRef Link)
[24] Calheiros, Rodrigo N., et al. "Workload prediction using ARIMA model and its impact on cloud applications' QoS." IEEE Transactions on Cloud Computing 3.4 (2014): 449-458. Article (CrossRef Link)
[25] Y.-C. Chang, R.-S. Chang, and F.-W. Chuang, "A predictive method for workload forecasting in the cloud environment," in Advanced Technologies, Embedded and Multimedia for Human-Centric Computing, pp. 577-585, Springer, 2014. Article (CrossRef Link)
[26] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "Rvlbpnn: A workload forecasting model for smart cloud computing," Scientific Programming, vol. 2016, 2016. Article (CrossRef Link)

Alanazi Rayan received his BSc in Computer Engineering from Dankook National University (DKU), Korea, in 2013, and an MSc in Computer Science from DKU in 2015. He received his PhD in Computer Engineering in the Database Lab at DKU in August 2019. He works as a lecturer in the College of Computer Science and Information Technology at Jouf University, Saudi Arabia.

Muhammad Ashfaq Khan is a PhD student at Dongguk University, Seoul, South Korea. His research interests include big data analytics, artificial intelligence, machine learning, deep learning, cloud computing, and computer networks. He received a Master of Engineering in communication systems & networks from Mehran University of Engineering & Technology, Jamshoro, Pakistan.

Fawaz Alhazemi received a BSc in Computer Engineering from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 2003. Also, he received an MSc in Information and Communications Engineering in 2010 and a PhD in Electrical Engineering in 2019 from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. He is currently working as an Assistant Professor in the Department of Computer and Network Engineering at the College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia. He has authored or co-authored over 35

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)

technical papers. He is a Technical Program Committee member of several international conferences and serves as a Reviewer for IEEE Transactions/Magazines and International Journals. His research interests are green data centers and energy-aware computing. He received the Best Paper Award and the Best Student Paper Award at FTRA AIM-2014 and at the 6th CloudComp-2015, respectively. He is an IEEE Senior Member and ACM Senior Member.

Hamoud Alshammari received a BSc in Computer Information Systems from King Saud University, Saudi Arabia, in 2002. He received his MBA from Yarmok University, Jordan. Then, he received an MSc in Computer Science from the University of Bridgeport, CT, USA. He is doing his PhD in Computer Science and Engineering at the University of Bridgeport. He is doing research in Big Data and Hadoop MapReduce performance. He is also interested in data analysis. Mr. Alshammari is a member of the Upsilon Pi Epsilon Honor Society.

Yunmook Nah received his BSc in Computer Engineering from Seoul National University (SNU), Korea, in 1986, and an MSc in Computer Engineering, Database, from SNU in 1988. He received his PhD in Computer Engineering, Database, from SNU in 1993. Currently, he is a Full Professor in the Applied Computer Engineering Department at Dankook University.

Copyrights © 2020 The Institute of Electronics and Information Engineers

이화여자대학교 | IP:203.255.***.68 | Accessed 2020/05/18 03:52(KST)