Workload Prediction for Cloud Cluster using A Recurrent Neural Network

Weishan Zhang, Bo Li, Dehai Zhao, Faming Gong, and Qinghua Lu
School of Computer and Communication Engineering, China University of Petroleum
No.66 Changjiang West Road, Qingdao, China

Abstract

Maximizing benefits from a cloud cluster with minimum computational costs is challenging. An accurate prediction to cloud workload is important to maximize resources usage in the cloud environment. In this paper, we propose an approach using recurrent neural networks (RNN) to realize workload prediction, where CPU and RAM metrics are used to evaluate the performance of the proposed approach. In order to obtain optimized parameter set, an orthogonal experimental design is conducted to find the most influential parameters in RNN. The experiments with Google Cloud Trace data set shows that the RNN based approach can achieve high accuracy of workload prediction, which lays a good foundation for optimizing the running of a cloud computing environment.

Keywords

Cloud cluster, Recurrent neural networks, Workload prediction

I. Introduction

Optimal running of Cloud Computing environment is essential to provide services with high dependability. An accurate cloud cluster workload prediction is helpful to improve the resource allocation strategy and adapting to different business requests.

In literature, there has been some work on hostload prediction [1], task usage pattern [2] and characterizing task constraints [3]. However, there is scarce work on predicting Cloud resource requests. On the other hand, most of these existing work [1], [4], [5], [6], [7], [8] use linear or probabilistic models with hand-crafted features which are laborious and mostly dependent on domain knowledge and experiences. This may cause inappropriate prediction patterns [1].The work in [9] has proved that it can use internal memory units to solve sequence analysis problems. Zachary C. Lipton[10] showed that non-linear activation RNN model has a strong sequence learning capability.

As the resource requests in a cloud computing environment can be considered as a time sequence, in this paper, we propose to take advantages of RNN's capabilities to predicted cloud resource requests. Statistics based approach, specifically an orthogonal experimental design(OED) is used to find the most influential parameters in a RNN network.

The rest of this paper is organized as follows. In the second section, we introduce basic concepts of RNN, and present our approach on resource request prediction using RNN. The third section showed our experiments, including an introduction of the dataset we used, data preprocessing, experimental design, parameters optimization, prediction accuracy assessment and results analyzation. Conclusion and future work end the paper.

II. Basic Introduction to Recurrent Neural Networks

As shown in Figure 1, a RNN mainly consists of three parts:

- The input layer is $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t)$, where $\boldsymbol{x}_t = (x_{t1}, x_{t2}, ..., x_{tn})$. t is the time step and n is the number of input nodes. So the input of each training cycle is a two-dimensional matrix. The first dimension represents time and the second dimension represents metrics.
- The output layer is $\mathbf{Y} = \boldsymbol{y}_T^{'} = f(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t)$, where $\boldsymbol{y}_T^{'}$ is the output of the Tth input. The final output is a 2D matrix. And $\mathbf{Y} = f(\mathbf{X})$ can be understood as a simple non-linear function.
- The hidden layer completes the major function of RNN. Nodes in hidden layers are marked as $H = h_0, h_1, ..., h_t$ and they are memory cells of the network. Current messages can be transfered to the next step. Additionally, messages can be transfered from output layer to hidden layer because nodes in the hidden layer are connected to each other and the input of hidden layer includes the state of hidden layer. When inputing a time sequence to a trained RNN, it will output the predicted value of the next state $\boldsymbol{y}_T^{'}$. Then $\boldsymbol{y}_T^{'}$ is compared with the ground truth $\boldsymbol{y}_{T+1}$ to compute loss.

Statistical patterns of random data sequence in time series analysis can be used to solve practical problems. For example, workload state in the cloud cluster is a continuous time sequence of numbers, which can be used to predict trends of workload state. The sequence is random because of the impact of random factors. In RNN, in order to model time series, the neuron output with direct effect should go to itself in the next time stamp, and increase the interconnections between hidden layer nodes. Therefore $\boldsymbol{y}_T^{'}$(the final result of T time in RNN model) is the result of the common action of input and all historical parameters. Using RNN to establish mathematical models of curve fitting and parameter estimation can reflect data fluctuations effectively.

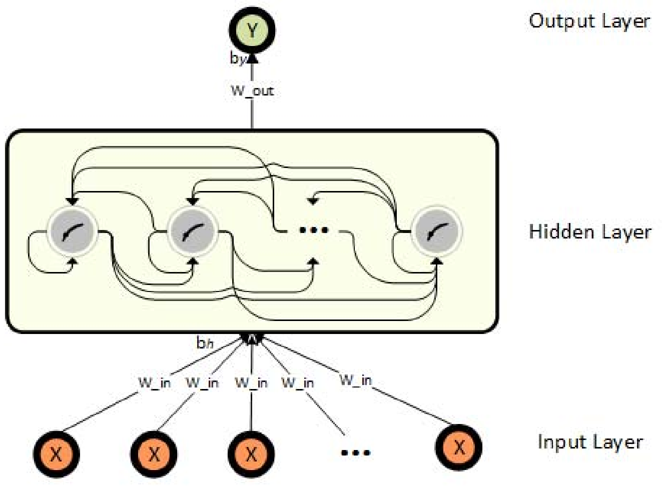

Figure 1. Structure of Recurrent Neural Networks

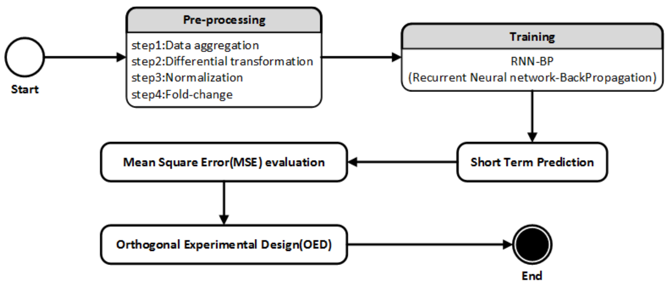
Start
Pre-processing
step1:Data aggregation
step2:Differential transformation
step3:Normalization
step4:Fold-change
Training
RNN-BP
(Recurrent Neural network-BackPropagation)
Short Term Prediction
Mean Square Error(MSE) evaluation
Orthogonal Experimental Design(OED)
End

Figure 2. Work flow of the RNNs-based approach

III. Overview of RNN based Workload Prediction

Figure 2 shows the work flow of RNN based approach.

A. Dataset and Data Preprocessing

The dataset we used is the Google cloud trace in 2011. It contains 672,075 jobs and more than 48 million tasks in 7000 heterogeneous machine server clusters over 29 days. These data are used to train a RNN model and predict the resource requests at the next period. Before this, we should conduct data preprocessing. It involves the following steps:

- Firstly, the resource request metrics are aggregated from the dataset, so that they can record the time stamp of each minute. Then the dataset is divided into training set and test set, and they contain 80% and 20% samples respectively.
- Secondly, as we knows, if there is a strong linear relationship between data, the learning ability of a artificial neural network will be weak. Therefore we reduced the linear degree of aggregated data using a differential transformation $x^d(t) = x(t) - x(t-d)$, where d is delay time and can be determined by the ARIMA (Autoregressive Integrated Moving Average Model) model[11].
- Thirdly, normalize the data after differential transformation

$$x_i^{'} = \frac{x_i}{max(x_i) - min(x_i)}$$

. Then the transformed data is normalized into the range (0, 1).
- The last step is folding every sample data to a 3D matrix.

$$\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$$

$$= \left[\begin{bmatrix} x_{11} & \dots & x_{1n} \\ \dots & & \dots \\ x_{t1} & \dots & x_{tn} \end{bmatrix} \dots \begin{bmatrix} x_{T1} & \dots & x_{Tn} \\ \dots & & \dots \\ x_{(t+T),1} & \dots & x_{(t+T),n} \end{bmatrix} \right]$$

B. Training RNN and Short-term Prediction

Comparing with traditional artificial neural networks, RNN shares parameters in the same layer. Nevertheless, RNN is similar with ANNs because they both use back propagation(BP) training process.

In the current work, we compare the training set after a short period of time sequence prediction results, therefore we call this short-term prediction. Suppose we take the training samples from the beginning of the data set at time $T(defaultinitialpositionofsamples)$, and we want to start prediction at some future time *t*, then the prediction interval should be $[t, t+s]$. In short term prediction our purpose is to investigate how the prediction accuracy changes with increments of s, how it differs with varying different factor levels. In our case, we commence prediction immediately after the end of the training set (i.e., $t = t_0$), and the prediction interval size s is the time step size of the RNN model to simplify the calculation.

C. Evaluation of Experimental Error

MSE (Mean Square Error) is used in the experiments to evaluate the performance of RNN.

$$MSE = \frac{\sum_{i=t}^{i=t+s}(l_i^{'} - l_i)^2}{s}$$

where $t = t_0$ is the end of the training set and the beginning of the test set. s is the next time unit of t_0 with an interval of $s = (nt, 2nt, 3nt)$. $l_i^{'}$ is the prediction of workload by the trained model, l_i is the ground truth. It is important to note that the prediction data must then be returned to their original format due to the transformations in the pre-processing.

D. Network Optimization Parameters

Orthogonal Experimental Design (OED) is a design method for solving problems with multiple levels and factors. According to orthogonality, this method selects a part of typical samples from overall samples to conduct experiment, which is an efficient and economical experimental design method. In our work, we use OED to find optimal parameters for RNN. Though it is difficult to identify whether the parameters of the RNN network fall into local optimal solution, OED tables can easily show which set of parameters is the best.

Generally, several factors such as input node, time step, output node, hidden node, learning rate and sample size will influence the performance of RNN. In our work, we only need to predict the next one state of input sample $x_{t^{'}}$, $t^{'} \in T$, so the number of output node should be 1. Chunyun Xu[12] et al. have confirmed that the combination of experience method and experimental method will be helpful to find a more stable state for a model. Therefore we changed the number of hidden nodes according to the number of input nodes.

$$m = \sqrt{n+1} + \alpha$$

m is the number of hidden nodes, n is number of input nodes, α is a constant between 1 to 10.

As shown in table I, we focused on the following 4 factors, the number of input node (n_in), the time step (n_step), the number of hidden node (n_hi) and learning rate (LR) to construct a $L_9(3^4)$ orthogonal table, in which every factor includes 3 levels.

Table I
LEVEL VALUES FOR EACH FACTOR

Levels	n_in	n_hi	n_step	LR
1	3	3	10	0.01
2	5	5	20	0.005
3	10	10	30	0.001

IV. Experimental Implementation and Evaluation

A. Orthogonal Table

Table II shows the parameters we used in our experiments under the guidance of $L_9(3^4)$. We conducted nine group of experiments to find the optimal parameters combination. There are 9 orthogonal experiments, each of which is assigned with one of each factor's levels, and so will identify the level set that generates the least prediction MSE.

B. Experimental results and analysis

Table III(a) is the prediction accuracy of Google cloud CPU requests trace. We can see that group 01 has the highest prediction accuracy with the lowest average MSE of 2.761×10^{-5}. The parameters of group 01 are set as 3 input nodes, 3 hidden nodes, 10 step length and 0.01 learning rate.

Table III(b) shows the prediction accuracy of Google cloud RAM requests trace. We can see that group 06 achieves a better result than other groups, whose average MSE is 1.2472×10^{-4}. The parameters of group 06 are set as 5 input nodes, 10 hidden nodes, 10 step length and 0.005 learning rate.

Table II
Level values for each factor

Num	n_in	n_hi	n_step	LR
01	3	3	10	0.01
02	3	5	20	0.005
03	3	10	30	0.001
04	5	3	20	0.001
05	5	5	30	0.01
06	5	10	10	0.005
07	10	3	30	0.005
08	10	5	10	0.001
09	10	10	20	0.01

Table III
OED for prediction accuracy of CPU requests AND OED for prediction accuracy of RAM requests

Num	time unit(s)	MSE(10^{-5})	AveMSE
01	1	3.1164	2.7610
	2	2.5592	
	3	2.6343	
02	1	2.8007	3.0917
	2	3.1291	
	3	3.3455	
03	1	3.0743	3.4839
	2	3.7324	
	3	3.6449	
04	1	4.4861	4.7758
	2	4.6999	
	3	5.1413	
05	1	5.1153	5.3191
	2	5.3696	
	3	5.4724	
06	1	3.8963	4.3508
	2	4.3775	
	3	4.7789	
07	1	4.5232	4.7032
	2	4.7551	
	3	4.8314	
08	1	4.2431	4.5325
	2	4.8824	
	3	4.4722	
09	1	4.9232	4.7409
	2	4.5068	
	3	4.7927	

Num	time unit(s)	MSE(10^{-5})	AveMSE
01	1	0.9905	1.977
	2	2.6058	
	3	2.3347	
02	1	2.6214	2.2153
	2	2.3072	
	3	1.7173	
03	1	4.0878	3.6816
	2	3.5081	
	3	3.4488	
04	1	3.1777	3.1896
	2	2.9421	
	3	3.4469	
05	1	1.6956	1.505
	2	1.4991	
	3	1.3232	
06	1	1.1555	1.2472
	2	1.0126	
	3	1.5736	
07	1	1.3997	1.2872
	2	1.2683	
	3	1.1935	
08	1	2.7098	2.8008
	2	2.9034	
	3	2.7892	
09	1	1.6928	1.4063
	2	1.2919	
	3	1.2342	

Table IV(b) shows the analysis of OED. Using analysis of variance on the OED, each of which is assigned a score, we can estimate each factor's effect. For a specific factor, k_i represents the average score of each level of the factor. R_i is the data range of the ith factor, which is the difference between the maximum and the minimum in the k_i of that factor, and represents the factor's importance. For a factor, a superior level (SL) is the level with the highest average score in the k_i, representing the level that achieves the best prediction performance, and we can find the all level of each factor through the table I. MF is the main influence factor rank, obtained by sorting R_i to identify the most influential factor and how much it affects prediction performance. The ith experiment score is described as follows.

$$score(i) = \frac{Max\{\sqrt{AverageMSE}\} - Min\{\sqrt{AverageMSE}\}}{\sqrt{AverageMSE(i)}}$$

From Table IV(b), we can see that the range of n_in1 is the biggest on CPU requests analysis, so it should be first controlled in the optimal level. Because the three main effect size of factor n_in is ordered as n_in1, n_in3, n_in2, therefore the best level of n_in is n_in1. In the same way, the second important factors are n_step, the best level is n_step1, the third is n_hi1 and LR2. Through the above analysis, the optimal parameters of the network for CPU prediction is shown at the 1th experiment of table III(a). Similarly, the optimal parameter for RAM prediction is shown at the 6th experiment of table III(b).

C. Discussion

According to the above evaluations, we have the following observations:

Different parameter combinations should be applied to different kinds of time sequence. For example, the parameters combination of 3 input nodes, 10 hidden nodes 10 step length and 0.01 learning rate performs well for the CPU trace prediction, while the parameters combination of 5 input nodes, 10 hidden nodes 10 step length and 0.005 learning rate performs well for the RAM trace prediction.

Comparing with ARIMA[11] based prediction method, RNN based approach has a higher accuracy with the lowest average MSE of 2.761×10^{-5}, which is 50% smaller than ARIMA based method.

Table IV
RESULTS ANALYZATION OF ORTHOGONAL EXPERIMENTAL DESIGN

CPU requests analyzation					
Num	n_in 1	n_hi 2	n_step 3	LR 4	Score
01	3	3	10	0.01	0.388
02	3	5	20	0.005	0.3667
03	3	10	30	0.001	0.3454
04	5	3	20	0.001	0.295
05	5	5	30	0.01	0.2795
06	5	10	10	0.005	0.3091
07	10	3	30	0.005	0.2973
08	10	5	10	0.001	0.3028
09	10	10	20	0.01	0.2961
k1	0.3667	0.3268	0.3333	0.3212	
k2	0.2945	0.3163	0.3192	0.3243	
k3	0.2987	0.3169	0.3074	0.3144	
R	0.0722	0.0099	0.0259	0.0099	
SL	n_in1	n_hi1	n_step1	LR2	
MF	1	3	2	3	

RAM requests analyzation					
Num	n_in 1	n_hi 2	n_step 3	LR 4	Score
01	3	3	10	0.01	0.5704
02	3	5	20	0.005	0.5388
03	3	10	30	0.001	0.418
04	5	3	20	0.001	0.449
05	5	5	30	0.01	0.6537
06	5	10	10	0.005	0.7181
07	10	3	30	0.005	0.7069
08	10	5	10	0.001	0.4792
09	10	10	20	0.01	0.6763
k1	0.509	0.5754	0.4295	0.6334	
k2	0.607	0.3975	0.5547	0.6546	
k3	0.461	0.6041	0.5928	0.289	
R	0.146	0.2066	0.1633	0.3656	
SL	n_in2	n_hi3	n_step3	LR2	
MF	4	2	3	1	

V. RELATED WORK

Xindi Cai[13] predicted 100 missing values from 5000 time sequence data points. The research showed that time sequence analysis requires models and techniques for chaotic systems. In their experiment, RNN is trained for time sequence prediction using Hybrid PSO-EA learning algorithm, which gives us a good reference for training the model.

A. Khan[14] realized workload characterization and prediction in cloud cluster, based on Hidden Markov Modeling (HMM). They characterized temporal correlations and used them to predict various workload patterns. R. N. Calheiros[15] proposed an ARIMA model to predict the future state of the workload. They evaluated effect of resource utilization and quality of service regarding the prediction accuracy. We use the latest technology and achieve a better accuracy.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we used Google cloud CPU and RAM trace to train RNN models and predicted the workload state in the cloud cluster. By analyzing the results of OED, we can find the optimal parameters combination. In addition, we found that different parameters combination should be applied to different kinds of time sequence. With the smallest average MSE of 2.761×10^{-5}, our method can predict workload state accurately. And the good performance showed that RNN based method is suitable for solving time sequence.

Though RNN base method performs well for time sequence prediction, it can only solve short-term time sequence. When facing long-term time sequence prediction task, RNN based method is not qualified. We can try some other methods such as LSTM[16], [17] or CW-RNNs[18] to solve this problem.

ACKNOWLEDGMENTS

This research was supported by the Program on Innovative Methods of Work from Ministry of Science and Technology, China (Grant No. 2015010300).

REFERENCES

[1] D. Sheng, D. Kondo, and W. Cirne, "Host load prediction in a google compute cloud with a bayesian model," in *IEEE/ACM 24th International Conference for High Performance Computing, Networking, Storage and Analysis (SC'12)*, 2012, p. 21.

[2] Q. Zhang, J. L. Hellerstein, and R. Boutaba, "Characterizing task usage shapes in google's compute clusters," in *Large Scale Distributed Systems and Middleware Workshop (LADIS'11)*, 2011.

[3] B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das, "Modeling and synthesizing task placement constraints in google compute clusters," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 3.

[4] S. Chen, M. Ghorbani, Y. Wang, P. Bogdan, and M. Pedram, "Trace-based analysis and prediction of cloud computing user behavior using the fractal modeling technique," in *Big Data (BigData Congress), 2014 IEEE International Congress on*. IEEE, 2014, pp. 733–739.

[5] 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.

[6] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 500–507.

[7] P. Saripalli, G. Kiran, R. R. Shankar, H. Narware, and N. Bindal, "Load prediction and hot spot detection models for autonomic cloud computing," in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*. IEEE, 2011, pp. 397–402.

[8] Y. Jiang, C.-s. Perng, T. Li, and R. Chang, "Asap: A self-adaptive prediction system for instant cloud resource demand provisioning," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 1104–1109.

[9] M. M. Botvinick and D. C. Plaut, "Short-term memory for serial order: a recurrent neural network model." *Psychological review*, vol. 113, no. 2, p. 201, 2006.

[10] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[11] E. S. Gardner Jr and E. McKenzie, "Notełseasonal exponential smoothing with damped trends," *Management Science*, vol. 35, no. 3, pp. 372–376, 1989.

[12] C. Xu and C. Xu, "Optimization analysis of dynamic sample number and hidden layer node number based on bp neural network," in *Proceedings of The Eighth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 2013*. Springer, 2013, pp. 687–695.

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

[14] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 1287–1294.

[15] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications qos," *Cloud Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 449–458, 2015.

[16] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *The Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

[17] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[18] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork rnn," *arXiv preprint arXiv:1402.3511*, 2014.