2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing

Unsupervised neural predictor to auto-administrate the cloud infrastructure

Hanen Chihi
University of Tunis
Higher Institute of Computer Science/SOIE
Tunis, Tunisia
e-mail: hanen.chihi@gmail.com

Walid Chainbi
University of Sousse
Sousse National School of Engineers/SOIE
Sousse, Tunisia
e-mail: Walid.Chainbi@gmail.com

Khaled Ghedira
University of Tunis/SOIE
Tunis, Tunisia
e-mail: Khaled.Ghedira@isg.rnu.tn

Abstract—**Due to all the pollutants generated by it and the steady increases in its rates, energy consumption has become a key issue. Cloud computing is an emerging model for distributed utility computing and is being considered as an attractive opportunity for saving energy through central management of computational resources. Obviously, a substantial reduction in energy consumption can be made by powering down servers when they are not in use. This work presents a resources provisioning approach based on an unsupervised predictor model in the form of an unsupervised, recurrent neural network based on a self-organizing map. Unsupervised learning in computers has for long been considered as the desired ambition of computer problems. Unlike conventional prediction-learning methods which assign credit by means of the difference between predicted and actual outcomes, the proposed study assigns credit by means of the difference between temporally successive predictions. We have shown that the proposed approach gives promising results.**

Keywords-component; Cloud computing; prediction; neural network, green computing.

I. INTRODUCTION

Cloud computing platforms are being increasingly utilized by industry, government and academia due to their ability to deliver robust and scalable computational power [1] [10]. These systems are dynamically provisioned based on a determination of the required computing resources requested by the end user of the cloud application [6] [7].

With dramatically increasing demand on computing and storage systems, cloud infrastructures have been scaled tremendously which results in huge amount of energy consumption, heat dissemination and even part of climate change. Lack of energy-conscious provisioning techniques may lead to overheating of cloud resources in case of high loads. This in turn, may result in reduced system reliability. Another related issue is the carbon dioxide emission that is detrimental for the physical environment due to its contribution in the greenhouse effect. All these problems require the development of efficient strategy. As such, green computing has come to the picture seeking solutions for cloud infrastructures to be energy efficient.

Devising methods for reducing power consumption and environmental impact through cloud auto-scaling is hard. Auto-scaling must ensure that resources can be provisioned quickly to meet QoS requirements as load changes. If auto-scaling responds to load fluctuations slowly, applications may experience a period of poor response time awaiting the allocation of additional computational resources. One way to mitigate this risk is to use neural network to predict in advance needed resources regarding historical. So predicted resources can be booted and configured in order to be allocated rapidly.

There has been a great deal of research on dynamic resource prediction and allocation for physical and virtual machines and clusters of virtual machines [9] [12]. Moreover, there were many studies devoted to autonomic configurations of VM parameters. For example, in [13], feedback control approaches had achieved notable successes in adaptive virtual resource allocation and Web application parameter tunings. However, such control approaches rely on explicit models of target systems. Other studies formulated the problem as a combinatorial optimization [8] [3].

Learning to predict is one of the most basic and prevalent kinds of learning. Most pattern recognition problems can be treated as prediction problems in which the classifier must predict the correct classifications. Learning to predict problems also arise in heuristic search. An important advantage of prediction learning is that its training examples can be taken directly from the temporal sequences of ordinary sensory input; no special supervisor or teacher is required [5] [11].

The increasing demand for cloud computing resources has led to a commensurate increase in the operating power consumption of the systems that comprise the cloud. In this paper, we present a novel approach of an unsupervised neural predictor based on a self-organizing map (RSOM). We introduce a class of incremental learning procedures specialized for prediction; that is, for using past experience with an incompletely known system to predict its future behavior. Unlike conventional prediction-learning methods which assign credit by means of the difference between predicted and actual outcomes, the proposed study assigns credit by means of the difference between temporally successive predictions.

The novelty of our solution lies in its integration of an unsupervised neural predictor and the development of policies for dynamic provisioning of additional servers in the cloud, as well as considering server's timing requirements.

The rest of this paper is organized as follows. Section 2 presents the RSOM. Section 3 describes the problem and the system model. Section 3 presents experimental results and case study. Section 4 deals with related work and draws comparisons between our method and other methods. Finally, we conclude our study in section 6.

II. Recurrent self-organizing map

The Self-Organizing Map (SOM), introduced by Kohonen, is a neural network (NN) based on an unsupervised competitive learning [15]. We concentrate on the SOM which has a feed-forward structure with a single computational layer of neurons arranged in rows and columns. Each neuron is fully connected to all the source units in the input layer. Figure 1 shows a bidimensional SOM.

We present as an extension to the SOM, the RSOM [14] that allows storing certain information from the past input vectors (see Figure 2). The information is stored in the form of difference vectors in the map units. The mapping that is formed during training has the topology preservation characteristic of the SOM.

In the recurrent self-organising map neural architecture, the inputs to the model at each time-step are represented by a moving window and the activations on the RSOM output layer for the previous time-steps. By using these inputs and their associated learned weights, the RSOM architecture creates a topological temporal representation on the output layer for each input slice.

In the training algorithm (Figure 2), an episode of consecutive input vectors *x(n)* starting from a random point in the input space is presented to the map. The difference vector $y_i(n)$ in each neuron of the map is updated as follows:

$$y_i(n) = (1-\alpha)y_i(n-1) + \alpha(x(n) - m_i(n)) \quad (1)$$

where $m_i(n)$ is the weight vector, *x(n)* is the input pattern, $0<\alpha<1$ is the memory coefficient and $y_i(n)$ is the leaked difference vector for the neuron *i* at step *n* while large *α* corresponds to short term memory whereas small values describe long term memory, or a slow activation decay. Each neuron involves an exponentially weighted linear recurrent filter with the impulse response. At the end of the episode (step n), the best matching unit (BMU) b is searched by:

$$y_b = \min_{i\in V} \|y_i(n)\| \quad (2)$$

where V is the set of all neurons comprising the RSOM. Weights are adjusted according to:

$$W_i(t+1) = W_i(t) + h_{bmu_i}(t)y_i(t) \quad (3)$$

where $h_{bmu_i}(t)$ is the value of the neighbourhood function.

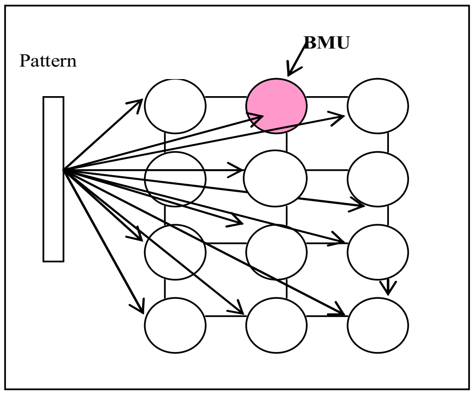

Figure 1. Bidmentional SOM.

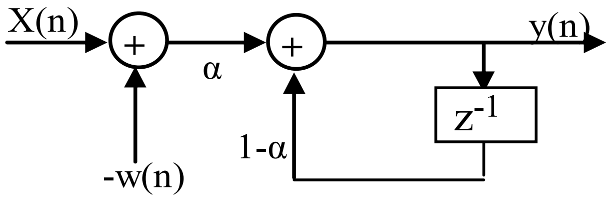

Figure 2. Learning process of RSOM.

III. Autonomic virtual resource prediction based on RSOM

In this section, we present the architecture of the proposed model, focusing on its components and their main requirements.

A. *Provisioning scenario*

Dynamic provisioning refers to the ability of dynamically acquiring new resources and integrating them into the existing infrastructure and software system. Figure 3 describes the common scenario in which dynamic provisioning constitute an effective solution for cloud.

Dynamic provisioning in our model is mostly focused on controlling the lifetime of virtual nodes where to deploy the users' applications. Hence, it specifically refers to the provisioning of hardware in the form of VM whether they are provided by an IaaS provider.

The neural predictor is in charge of making the provisioning happen by interacting with IaaS providers and local virtualization facilities. The proposed model provides the following functionalities:

- Requesting virtual machine instances from a variety of virtual machine managers or IaaS providers;
- Detecting dynamically the need of additional resources to maintain the desired quality of service for application execution using a neural predictor;
- Controlling the lease of dynamic resources to optimize their usage and eliminating resources that would otherwise remain idle and waste power using a neural predictor.

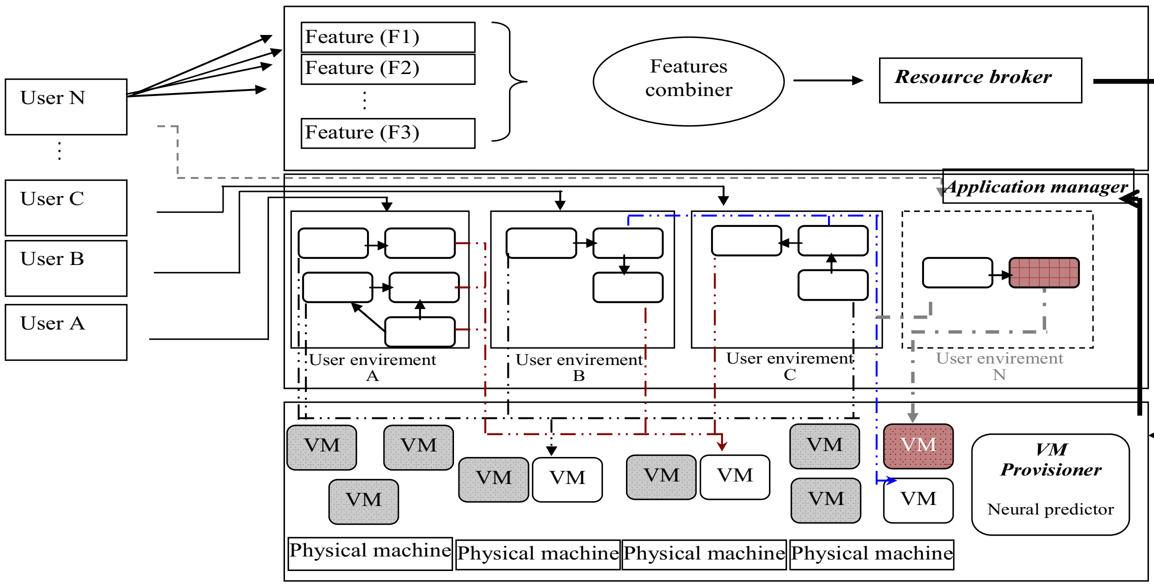

Figure 3. Structure of the proposed model.

The system is completely modular and can be fully customized to activate and control the behavior of dynamic provisioning. The proposed model integrates dynamic provision capabilities as part of the model and dynamic provisioning is the result of the collaboration of several components:

Resource Pool Monitor: The main responsibility of the resources pool monitor is controlling the life cycle of resources and predicting increases or decreases of the utilization of resources. The pool manager also notifies the provisioning service when a dynamic resource is activated and terminated.

Resource self-administration module: This service is responsible for satisfying a provisioning request. It can be considered as a black box that mainly performs the following operations: resource allocation, resource release, resource configuration, resource re-configuration, monitoring of the resource status and the resource pool status.

Resource predictor: The resource predictor encapsulates a neural predictor that uses the history of user requests for the prediction of future needs for resources.

B. Prediction model based RSOM

The prediction module starts by collecting historical load, and predict loads in the future based on the historical load. Depending on the predicted future load, the self-administration module dynamically adjusts resources allocations to minimize energy consumption. From this definition, we can represent the prediction module as follows:

- **Sensor**: It watches over the execution of the VM and the satisfaction of user requests. In addition, it collects the history of user demands in order to use them for neural network training.
- **Actor**: It is a neural network using an unsupervised learning and recurrent connections: the Recurrent Self-Organizing Map (RSOM). Its role is to determine the number of VM to add or to stop at time $t + \lambda$.
- **Effector**: According to the future charge provided from the RSOM, the effector sends the needed information to the resources pool monitor.

In order to make meaningful predictions, the RSOM needs to be trained on an appropriate data set. Basically, training is a process of determining the connection weights in the network. The final goal is to find the weights that minimize an overall error measure, such as the sum of squared errors or mean squared errors.

An important phase in the modeling of neural network is the construction of the training set and more precisely the structure of input vectors and output of the network. Representation of the user demand evolution, with respect to time and depending on the used pretreatment method, is an important step for learning neural networks. The pretreatment method should be less redundant than the original signal while preserving the discriminant information. The choice of a technical parameter is of fundamental importance because it determines the efficiency of neural prediction.

In this paper, we propose the following structure for the input vector:

- Number of VM used at time t. For each VM, we identify its characteristics (CPU, memory, bandwidth ...).
- The number of requests for this moment.
- At $t + \lambda$, RSOM predict the potential applications number and the VM needed to satisfy them.

Learning algorithm of RSOM for prediction is implemented as follows. An episode of consecutive input vectors starting from a random point in the data is presented to the map. The number of vectors belonging to the episode is dependent on the leaking coefficient α used in the units.

The best matching unit is selected at the end of the episode based on the norm of the difference vector. The updating of the vector and its neighbours is carried out as in the equations (1) and (3). After the updating, all difference vectors are set to zero, and a new random starting point from the series is selected. The above scenario is repeated until the mapping has formed. In prediction the best matching unit of RSOM is searched for each input vector. A local model that is associated with the best matching unit is then selected to be used in the prediction task at that time. Figure 4 details the training and decision model of the proposed prediction module.

IV. EXPERIMENTAL RESULTS

In order to verify the effectiveness of our prediction model based on RSOM networks, we have collected a part of the operating traces from web servers [17].

The used benchmark is formed by two traces containing two month's worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida [17].

To evaluate our approach, we study two other neural network training modes:

- Supervised static training mode using the Multi-layer Perceptron (MLP).
- Unsupervised static training mode using the SOM network.

A. Data analysis

Before training a neural network, we need to define the training data sets (input and output) and to configure the network. This section describes the collection and the pretreatment of the training data. The Input data are defined as follows:

Requests: the amount of requests at the time. The Unity used was requests per second;

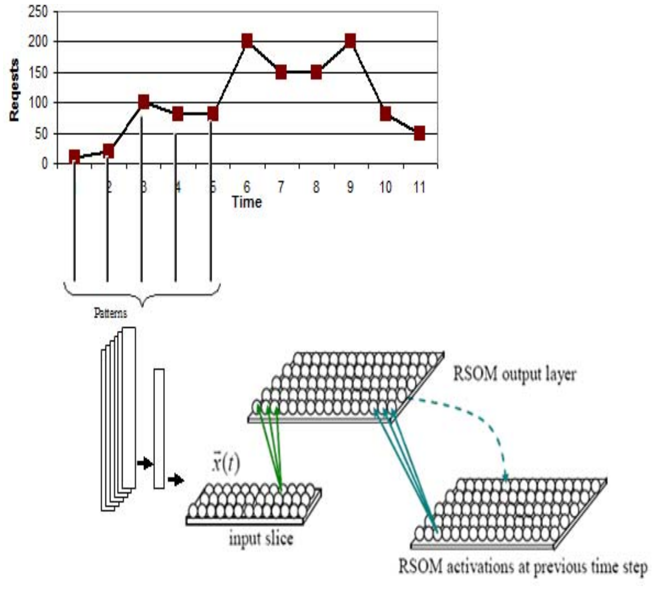

Figure 4. Neural predictor based RSOM.

- Time per request: time/second to complete request;
- Memory consumption: the amount of memory consumed by the requests;
- Processor load: the processor load is the capacity load from the system processor. The load represents the sum of all the processes that are being executed plus the waiting process queue;
- Storage: storage size consumed by the requests.

In the output, the network has to predict the resource provisioning needed by the environment, using the current context. These values are:

- Processor cores: to determine the number of cores necessary to allocate new VM;
- Memory: amount of memory necessary to carry out the current workload from the environment;
- Storage: amount of storage needed for the environment to maintain the current workload.

Using the network outputs, we can deduce the following information: the VM number and the datacenter number.

B. Setting of neural networks and discussion

To train the different neural network, we provided data that represented the current input and the desired output. To obtain these data, we prepared some Java programs to monitor the workload of user demands. With the aid of the collected data, we built neural networks using a Matlab and Neural Network toolbox. To test and evaluate the developed networks, we used Mean Squared Error (MSE) test measure, and after constructing a confusion matrix.

For the MLP neural network, the best suitable topology was formed by 5 input units, two hidden layers, with 7 and 12 units respectively, and 3 output units representing the predicted values. The used learning rate was 1e-2. With this configuration the MSE was 0.028 after 100 epochs.

After the training process of the SOM neural network, the best suitable topology was a rectangular map formed by 10x10 units, the training rate decrease linearly from 0.99 to 0.1 and the neighborhood radius decreases also linearly from the half of the diameter of the map to 1. With this configuration the MSE was 0.015 after 100 epochs.

We have implemented the RSOM neural network in order to simulate the learning algorithm presented in [19]. The development of the RSOM is a delicate task which requires many experiments. In fact, model performance depends on parameters chosen at the initialization and at the training phases. The training rate decrease linearly from 0.99 to 0.02 and the neighborhood radius decreases also linearly from the half of the diameter of the map to 1. The map size is set to 10x10. The term α corresponding to the moving window or the episode is set to 0.2. With this configuration the MSE was 0.00081 after 100 epochs.

There is no theoretical guidance on the choosing of parameters in prediction model based on neural network. Therefore, we achieve the chosen of parameters of prediction model based on each network through the program. In addition, we determine optimal parameters by comparing the MSE values given by different neural network structures that use different parameters.

For the MSE performance metric, MSE using MLP and SOM are relatively large, while using RSOM, the MSE value is relatively small. This implies that the RSOM prediction neural netwok has higher prediction accuracy in training and predicting the log data.

Moreover, with regard to the low MSE of SOM compared to MLP, we can note that the convergence of SOM occurs at a much faster rate than MLP in training mode. This is justified by the fact that the SOM uses unsupervised training algorithm. However, the MLP uses the backpropagation algorithm which is characterized by a long-term training.

Figure 5 represents the predicted and the real data generated respectively by the MLP, SOM and RSOM neural networks using training data. We can conclude that the users' demand trend forecasting of RSOM network based prediction model in Web type of cloud services, has higher prediction accuracy.

V. RELATED WORK

There has been a great deal of research on dynamic resource allocation for physical and virtual machines and clusters of virtual machines. In [13], Van et al. propose a two-level control loop in order to make resource allocation decisions. More precisely, the authors propose an autonomic resource management system which relies on two-level architecture with a clear separation between application specific functions and a generic global decision level. In their study, the authors do not address integrated management of a collection of physical machines. We resort to utility functions to map the current state of each application (workload, resource capacity, SLA) to a scalar value that quantifies the "satisfaction" of each application with regard to its performance goals. These utility functions are also the means of communication with the global decision layer which constructs a global utility function including resource management costs. They separate the VM provisioning stage from the VM placement stage within the global decision layer autonomic loop and formulate both problems as Constraint Satisfaction Problems (CSP).

Foster et al. [4] address the problem of deploying a cluster of virtual machines with given resource configurations across a set of physical machines.

Czajkowski et al. [3] define a Java API permitting developers to monitor and manage a cluster of Java VMs and to define resource allocation policies for such clusters.

Bolik et al. [2] present a statistical machine learning approach to predict system performance and minimize the number of resources required to maintain the performance of an application hosted on a cloud.

Duy et al. [5] propose aiming at designing, implementing and evaluating a Green Scheduler for reducing energy consumption of datacenters in Cloud computing platforms. It is composed of four algorithms: prediction, ON/OFF, task scheduling, and evaluation algorithms. The prediction algorithm employs a neural predictor to predict future load demand based on historical demand. According to the prediction, the ON/OFF algorithm dynamically adjusts server allocations to minimize the number of servers running, thus minimizing the energy use at the points of consumption to benefit all other levels.

Shi et al. [11] propose a prediction model based on RBF (Radial Basis Function) neural network, which is used to predict end-users resource demand in advance. The prediction of dynamic resource demand is an important basis for decision making of multi-scale resource elastic binding. Realizing multi-scale resources elastic binding in a controllable manner through predicting resources demand of end-users is critical for the rapid and elastic resources provision for cloud services.

Our research is different from current typical works in many aspects. It is based on auto-adapting the provisioning process while maintaining the performance and reducing the cost of the infrastructure. It is also different from proactive prediction and dynamic resources provision in [3] [4], as well as statistical machine learning approach to predict in [2]. The different starting points of our research problem directly influence the design philosophy. Moreover, our prediction object is the end-user resource demand of cloud applications and the user number, user upload capacity and server bandwidth. It is different from the computing resources intra data centers in [4]. All of them did not answer whether neural networks and time series analysis are fit for predicting different kinds of resources provisioning.

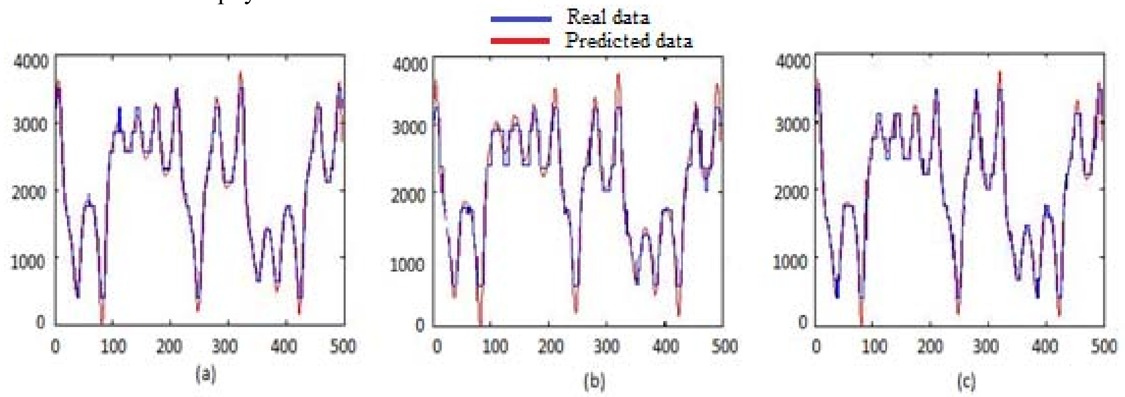

Figure 5. Predicted and the real data generated respectively by the MLP, SOM and RSOM neural networks.

Both models in [5] and [11] offer neural predictors: the first one is based on a Back propagated (BP) predictor and the second one is based on a RBF network.

Given a predicting object, it is difficult to determine what kind of training and learning method can achieve the fastest prediction speed and highest prediction accuracy. Since, there are many influencing factors, including the potential statistics law of the given data, the complexity of the problem, the number of samples and the predicted accuracy metrics etc. BP neural network represent an important type of prediction model. BP network mainly uses negative gradient descent method to tune weight and realize function approximation. However, there is lack of effective method in effectively choosing BP learning rate and deciding the number of hidden neurons. Furthermore, the method of tuning BP weight has obvious limitations: slow convergence and local minimum. Compared with BP, RBF is better than BP in many aspects. For example, the learning time is faster than BP. Learning rate and prediction speed have close relationship with the efficiency and effectiveness of multi-scale resource binding in cloud.

All the networks described above do not contain internal dynamics. In order to use these networks for the identification of nonlinear dynamic systems external dynamic elements are necessary. The networks having internal dynamics like dynamic multi-layer perceptrons and recurrent networks have also been tried for prediction purposes. This is an important reason why we choose RSOM.

Unsupervised learning of the temporal context is another attractive property of RSOM. It allows building models using large amount of data with only a little a priori knowledge. This property also allows using RSOM as an explorative tool to find statistical temporal dependencies from the process under consideration.

Most recurrent neural networks are trained via supervised learning rules. Only quite rare even no unsupervised neural networks models have been proposed for resource prediction, although, it can be argued that in temporal sequence analysis unsupervised neural networks could reveal useful information from the temporal sequences at hand in analogy to unsupervised neural networks' reported power in prediction problem.

By studying the two kinds of neural networks, we think RSOM network is an effective method to predict resource for end-users demand in resolving the problem of multi-scale resources elastic binding.

VI. Conclusion

This work proposes an autonomic resources provisioning for energy saving in the cloud. The proposed model computes the number of resources needed to process the load. The novelty of our solution lies in its integration of an unsupervised neural predictor and the development of policies for dynamic provisioning of additional servers in the cloud infrastructure, as well as considering server's timing requirements. Applying new computing paradigms, such as cloud computing with auto-scaling and autonomic provisioning, to increase server utilization and decrease idle time is therefore paramount to creating greener computing environments with reduced power consumption and emissions. We have shown that the proposed approach has provided promising results. These efforts have demonstrated both the feasibility and promise of autonomic resources provisioning for cloud environment.

In future work, we plan to implement a prototype that simulates the prediction of resources allocation in the cloud. Another plan is to compare the proposed model with other resources management schemes which employ different load prediction mechanisms.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patternson, A., Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Communications of ACM, vol. 53, no. 4, pp. 50-58, 2010.

[2] P. Bodik, R. Griffith, C. Sutton, A. Fox, M. Jordan, and D. Patterson, "Statistical machine learning makes automatic control practical for internet datacenters", in HotCloud'09: Proceedings of the Workshop on Hotp Topics in Cloud Computnig, 2009.

[3] G. Czajkowski, M. Wegiel, L. Daynes, K. Palacz, M.Jordan, G. Skinner, and C. Bryce, "Resource management for clusters of virtual machines", Cardiff, UK, 382–389, May 2005.

[4] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang,, "Virtual clusters for grid communities", Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium, Singapore, 513 - 520, May 2006.

[5] T.V.T. Duy, Y. Sato, and Y. Inoguchi, "A Prediction-Based Green Scheduler for Datacenters in Clouds", IEICE TRANS. INF. & SYST., vol.E94–D, no.**9**, 1731-1741, 2011.

[6] B. Golden, *Virtualization For Dummies*, vol.1, no.3, December 2007.

[7] J. Hurwitz, R. Bloor, M. Kaufman, and F. Halper, *Cloud Computing For Dummies*, no.**2**, November 2009.

[8] W. Iqbal, M.N. Dailey, D. Carrerab, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud", Future Generation Computer Systems, vol 27, 2011, 871–879.

[9] S. Kamboj, T. Estrada, M. Taufer and K.S. Decker, "Applying Organizational Self-Design to a Real-world Volunteer Computing System", AAMAS workshop on Agent Design: Advancing from Practice to Theory, 2009.

[10] P. Mell, and T. Grance, The NIST Definition of Cloud Computing (Draft), National Institute of Standards and Technology, vol.53, no.**6**, 1-7 ,2011.

[11] P. Shi, H. Wang, B. Ding, R. Liu, and T. Wang, "The Prediction Model Based on RBF Network in Achieving Elastic Cloud", AISS: Advances in Information Sciences and Service Sciences, Vol. **3**, No. 11, 67-78, 2011.

[12] B. Solomon, D. Ionescu, M. Litoiu, and G. Iszlai, "Designing autonomic management systems for cloud computing", International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI) , 631 – 636, 2010.

[13] H.N. Van, F.D. Tran, and J.M Menaud, "Autonomic virtual resource management for service hosting platforms", Workshop on Software Engineering Challenges in Cloud Computing, Vancouver, Canada : Canada, 1-8, 2009.

[14] M. Varsta, J. Del R. Millan, and J. Heikkonen, "A recurrent Self-Organizing Map for temporal sequence processing", In Proc. 7th Int Conf Artificial Neural Net-works, ICANN'97. Springer Verlag, 421-426, 1997.

[15] M. Varsta, J. Heikkonnen, and J.D.R. Millan, "ContextLearning with the Self-Organizing Map", Proceedings of Workshop on Self-Organizing Maps, Helsinki University of Technology, pp. 197-202, 1998.