Comparison of a Multi output Adaptative Neuro-Fuzzy Inference System (MANFIS) and Multi Layer Perceptron (MLP) in Cloud Computing Provisioning

Carlos Oberdan Rolim[1], Fernando Schubert[2], Anubis G. M. Rossetto[3], Valderi R. Q. Leithardt[1], Cláudio F. R. Geyer[1], Carlos B. Westphall[2]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[2]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC) – Caixa Postal 476 – 91.501-970 – SC – Brazil

[3]Instituto Federal Sul-Rio-Grandense – Campus Passo Fundo – Passo Fundo – RS – Brazil

{carlos.oberdan,valderi.quietinho,geyer}@inf.ufrgs.br,
{schubert,westphall}@inf.ufsc.br,
anubis.rossetto@passofundo.ifsul.edu.br

Abstract. *Cloud computing has changed the way that computing is delivered and used, turning it into a utility like water or electricity. In this context, many challenges and opportunities appear to make the Cloud a stable, accessible and trustworthy environment. Resource provisioning in the Cloud must be dynamic and able to adapt to changing needs. In this paper, a provisioning method is proposed that uses neural networks to provide the desired quality of service and assure SLA.*

1. Introduction

Throughout the history of computing, there have been several paradigm shifts from main-frames to mini computing to microprocessing to networked computers. Cloud computing is on track to be the next major paradigm shift. While the precise concept is still being defined, basically, cloud computing can be defined as a major advance in delivering services where information is stored and processed on the Internet (i.e., “the Cloud") usually via massive large-scale data centers which can be accessed remotely through various clients and platforms [Grimes, J., Jaeger, P. and Lin, J. 2008]. Essentially, cloud computing is a key concept that seeks to encapsulate the concept of ubiquitous processing and storage, by concealing the real complexity and underlying layers from the users [Grimes, J., Jaeger, P. and Lin, J. 2008].

In this paper specifically, we are addressing the following research questions, while providing the guidelines for an innovative and significant solution: (a) What kind of model is required to provide support for prediction and resource provisioning? (b) What tools need to be used for this purpose?

We argue that a model that applies a hybrid neural network to infer the current situation and predict future events can be used to overcome previous problems. Thus, in this paper our goal is to analyze the use of Multioutput Adaptative Neuro-Fuzzy Inference systems (MANFIS) as an underlying mechanism for predicting a desired model. To achieve this, we make a comparison between a MANFIS and a Multi Layer Perceptron (MLP) network with the back-propagation learning rule.

This paper examines advances in the state of the art by offering an alternative that combines a hybrid type of neural networks with wide-area distributed systems. This combination in resource provisioning for cloud computing has an innovative feature that can be exploited in a more satisfactory way. We stress that our proposal has focus on "low level" attributes to an underlying mechanism for predicting, we will not consider "high level" ones related to business impact and economics relations like monetary costs.

This paper is structured as follows: in Section 2 we comment on some related work. In Section 3 we present some alternative methods for dynamic resource provisioning and show why they are unsuitable to the needs of cloud computing. In Section 4 we introduce concepts used in the neural network and explain the reasons underlying the use of MANFIS. The standard approach to construct both Neural Networks and the experimental results are examined in Section 5. Finally some concluding remarks, known limitations and recommendations for future work are discussed in Section 6.

2. Related Work

Cloud Computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased [Armbrust, M., et. al 2009].

As recent works in Cloud area related to prediction we can found [Zohar, E. and Cidon, I. 2011] that presents a solution called Predictive ACK (PACK) for prediction in Traffic Redundancy Elimination (TRE). Although it is an excellent wok, it is for reducing bandwidth costs and not for prediction and resource provisioning. Another work is [Mallick, S. and Hains,G. 2011] that presents a cloud monitoring method based on prediction. It uses a rules-based system for monitoring and alert. The authors are working to use a mathematical model like quantitative analysis or Markov chain. We believe that an interesting alternative would be Neural Networks as mathematical model. Related to Neural Networks for prediction we can found several works in various areas of application. However, there are few works that uses a hybrid approach like ANFIS and MANFIS. We can found [Yilmaz, I. and Kaynar, O. 2011] that apply ANFIS for prediction of swell potential of clayey soils, [Elabd, S. and Schlenkhoff A. 2009] for travel time prediction and [Pramanika, N. and Kumar, R., 2009] that used a prediction model for water reservoir management. Those uses MANFIS we can cite [Gomathi, V., Ramar, K. and Jeevakumar, A. S. 2009] for human facial expression recognition and [Zhang, J., Chung, H.S. and Lo, W. 2008] for Chaotic Time Series Prediction. Regardless of be used in different scenarios, such works demonstrate the potential and flexibility of Neural Networks.

We point a lack of a solution that use a hybrid neural network as approach to provide certain SLAs and quality of service to their users and allow they to define the strict requirements and QoS attributes needed for Cloud systems to run smoothly. The alternatives to overcome this issue will be dealt with in the next sections and a proposal to address this problem will be outlined.

3. Alternatives for Dynamic Resources Provisioning

Compliance with the non-functional requirements that constitute the SLA requires a mechanism that is able to monitor and analyze the current state of the instance (or service) based on certain attributes (the desired granularity) and compare these results with the SLA agreed between the provider and the consumer.

On the basis of the analysis conducted in Section 2, most cloud computing solutions have some kind of SLA mechanisms and QoS, but these mechanisms are unable to represent the granularity and specific needs of consumers, for example, for enterprise mission-critical virtual instances.

Table 1. QoS Attributes

Requirements	*Specifications*	*Examples*
Adaptability	Allow dynamic reconfiguration during running time	Change in status from the requirements needed to instantiate a new virtual machine
Supervision	Monitor the current QoS	Monitoring parameters like CPU load, memory consumption
Prediction	Capacity to predict future behavior based on current historical data	Analyze the CPU load for a time and predict if more CPU cores are needed
Granularity	Allow the definition of specific parameters for each service	Define requirements during execution time such as response time and throughput

Granularity is the ability to ensure the availability and reliability of data and services hosted in the Cloud. This granularity corresponds to the typical non-functional requirements for each application or hosted service and depends on the perceived needs of each customer and consumer in the Cloud.

The primary need for the definition of such an algorithm or technique is that it meets the requirements of QoS in the context of cloud computing. Table 1 shows the proposed QoS requirements necessary for the dynamic provisioning mechanisms of the Cloud. These requirements have been drawn from the SLA requirements, as pointed out by [Aib, I. and Daheb, B. 2007], and the SLA requirements for IP networks, which can be applied to cloud computing.

Table 2. Analysis of algorithms and QoS techniques

Technique / Algorithm	*Adaptability*	*Supervision*	*Granularity*	*Prediction*
FIFO	No	No	No	No
PQ	No	No	No	No
CQ	No	No	No	No
WFQ	No	No	No	No
CBWFQ	No	No	No	No
RED	No	Yes	No	Yes
WRED	No	Yes	No	Yes
FRED	No	Yes	No	Yes
Architectural Transluscency	Yes	Yes	No	Yes

Technique / Algorithm	*Adaptability*	*Supervision*	*Granularity*	*Prediction*
Neural Networks	Yes	Yes	Yes	Yes

The survey of the QoS algorithms in Table 2 was conducted with the aim of seeking alternatives that are already consolidated and stable and that can fit the cloud computing model. The QoS algorithms used for network management, congestion, flow control and traffic prioritization have been evaluated [Aib, I. and Daheb, B. 2007]. In addition to these, solutions were sought such as those used in grid computing as well as Cloud computing proposals [Balen, D. and Westphall, C. B. 2011].

We argue that Neural networks can be applied to the Cloud model as a prediction and provisioning device. It were chosen because neural networks have the ability to learn from past behavior and predict future behavior on the basis of historical data.

4. Neural Networks with Clouds

Before constructing a model that can be implemented to show the provisioning, monitoring and prediction capabilities of cloud computing, we need to define its underlying mechanism. As argued above, the best choice is neural network. Neural network is a wide study and research field, with different network types and algorithms.

A commonly used Neural Network is Multi-Layer Perceptron (MLP). MLP consists of three or more layers (an input and output layer with one or more hidden layers). Each node implements a nonlinear activation function and an associated weight w_{ij} that is used to calculate the activation potential of the node. In MLP the calculation of error in the output of the network is given by the sum of squared errors for instant output of each node in the network and is represented by

$$E(n) = \frac{1}{2}\sum_{k=1}^{M} (d_k(n) - y_k(n))^2$$

For weight actualization, the method of steepest descent is used to update neurons in output and hidden layers. This method is known as the back-propagation learning rule. The configuration steps of MLP, involves defining the number of layers, the number of neurons in each layer, the activation function used in each layer, the learning rate and the number of epochs used to train the network according to the input data and desired output. These steps are laborious, require several analyze and adjustments to ensure the generalization power of network and due to the “black box” model of MLP, their internal functionalities are no “human friendly”.

An alternative means of overcoming these problems is having a network that “learns” about inferred data and represents this knowledge in a more “human” way. The most widely used approach is to combine fuzzy logic with neural networks to build hybrid networks. Different models can be employed to implement a Mamdani and Takagi Sugeno fuzzy inference systems like FALCON, ANFIS, NEFCON, NEFCLASS, NEFPROX, FUN, SONFIN, EFuNN and many others [Abraham, A. 2005]. In general Takagi-Sugeno has a lower Root Mean Square Error (RMSE) and produces a more accurate system than the Mamdani-type, which is much faster than Takagi-Sugeno [Marza, V. and Teshnehlab, M. 2009]. However, as we are seeking for accuracy, we did not consider models like FALCON, NEFCON, NEFCLASS and EfuNN, or even FUN that does not use a formal learning technique (it randomizes the parameter values of membership functions). With regard to other models, [Mackey, M.C. and Glass, L. 1977] provided a comparative performance of some neuro-fuzzy systems for prediction of chaotic time series (Table 3).

Table 3. Performance of some neuro-fuzzy models

System	*Epochs*	*Test RMSE*
ANFIS	75	0.0017
NEFPROX	216	0.332
EfuNN	1	0.0140
dmEFuNN	1	0.0042
SONFIN	1	0.0180

As shown, ANFIS has a lower RMSE than NEFPROX, SOFIN and dmEFuNN which uses the Takagi-Sugeno model. Thus our choice was to use ANFIS for prediction. However as our network needs to predict several values for resource provisioning, ANFIS cannot be used because it produces a single output. For this reason, we decided to use a Multi output Adaptative Neuro-Fuzzy Inference System [Benmiloud, T. 2010] which is an extension of the Adaptative Neuro-Fuzzy Network (ANFIS) [Jang, J. R. 1993]. As its name implies, MANFIS acts like multiple interconnected ANFIS providing multiple outputs instead of just one. As in the case of ANFIS, the strength of MANFIS lies in its ability to construct input-output mapping based on both human knowledge (in the form of fuzzy if-then rules) and learning the provided data.

Figure 1 shows a typical MANFIS with one-input for a first-order Sugeno fuzzy model with three outputs.

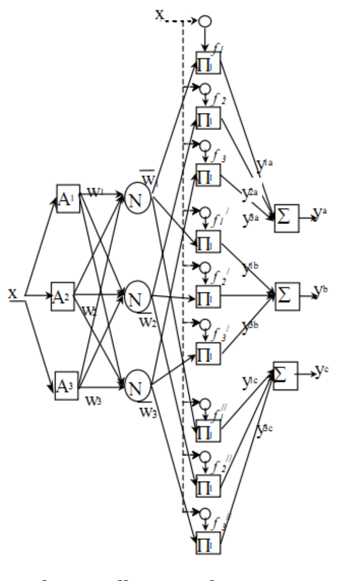

Figure 1: MANFIS for a one-input first-order Sugeno model with three rules – architecture with three outputs

In a MANFIS, each layer has a defined function: layer 1 maps the input to the fuzzy rules and its correspondent Membership Function (MF); in layer 2 the input signals are multiplied and the output is the product representing the firing strength of a rule; in layer 3 each rule´s firing strength is normalized; in layer 4 each node is an adaptive node that has a function with parameters defined by the output of layer 3; and in layer 5 the output is generated by the summation of all incoming signals.

What is interesting about MANFIS is that each network neuron implements a fuzzy set. The network can adjust the MF used by its internal fuzzy inference based on the data provided in the training phase. To adjust the MF parameters a gradient vector is used which measures the system inference by means of a set of rules that seek to reduce the global error. The adjustment of weights is done by using a two-passes hybrid algorithm that combines the Least-Square Estimator (LSE) with the Gradient Descent (GD) method. This hybrid learning approach is beneficial due to the fact that the convergence is much faster since the dimensions of the search space is reduced in the original pure backpropagation method used in ANN technique. In the case of MANFIS, this algorithm has to be adapted to work with multiple outputs instead of one, as with the original ANFIS. The backward pass is identical, the signal of error is backpropagated and the local parameters are updated by the gradient descent method. For a system with one output y:

$$a_{ij}(t+1) = a_{ij}(t) - \frac{h}{p}\left(\frac{\partial E}{\partial a_{ij}}\right)$$

where

h: the training rate for i_a and p: number of data of x (or y_d)

The partial derivative is used to update the parameters of the membership functions. The difference resides in forward pass, where the node output goes forward until layer 4 and the resulting parameters are identified and corrected by the sum of gradient of the errors of overall outputs (instead of just one error signal used by the least-squares method with one output). That is:

$$e(n) = x(n) - y_k(n)$$

$$a_{ij}(t+1) = a_{ij}(t+1) - \frac{h}{p}\left(\frac{\partial E_1}{\partial a_{ij}} + \dots + \frac{\partial E_n}{\partial a_{ij}}\right) \quad \text{which} \quad \frac{\partial E_n}{\partial a_{ij}} = f(e_n)$$

Following this explanation of the factors that determined the use of MLP and MANFIS, in the next section we will present the used approach to decide which one is most suitable for our Cloud environment and then analyze the experimental results.

5. Experimental Results

The developing an application in neural networks should be undertaken in the following stages: defining input and output data, configuring the network, training, testing and validation. With this in mind, as in the case of input data, we will provide the QoS attributes and system state variables used in Cloud environment. These inputs have been selected because of their strong influence on performance and system availability.

They are defined as follows: **(i_1)** *Requests*: the amount of requests at the time. The Unity used was requests per second (RPS); **(i_2)** *Concurrent Requests*: the amount of simultaneous requests that the environment receives; **(i_3)** *Time per request*: number of seconds for each request to be completed; **(i_4)** *Memory consumption*: the amount of

RAM consumed by the requests in a certain snapshot of time; **(i_5)** *Processor Load*: the processor load is the capacity load from the system processor. The processes are kept in wait or execution state. The load represents the sum of all the processes that are being executed plus the waiting process queue. The load may vary from 0 (empty queue) to 1.0 (full queue) and > 1.0 (full queue plus waiting processes); **(i_6)** *Storage Consumption*: Number of Megabytes consumption from the requests. In the output, the network has to predict the resource provisioning needed by the environment, using the current context. These values are: **(o_1)** Processor Cores: to determine the number of cores necessary to allocate a new virtual instance or update [one] a single running instance; **(o_2)** Memory: amount of memory necessary to carry out the current workload from the environment; **(o_3)** Storage: amount of storage needed for the environment to maintain the current workload. These inputs ($i_{1..6}$) and outputs parameters ($o_{1..3}$) was used in MLP and MANFIS.

Our first analysis was a MLP. To train the network, we provided data that represented the current input and the desired output (a Boolean value to indicate if the operational SLA was violated, the number of processor cores, the amount of memory and amount of storage provisioning required). To obtain these data, we prepared some scripts in PHP, Shell and Perl language to monitor the workload of a web server in different situations. The observed range was 30 days, which we believe represents different situations of use.

With the aid of the collected data, we built a MLP using a Matlab and Neural Network toolbox. To test and evaluate the generalization power of the developed network, we used the 10-fold cross-validation method, and after constructing a confusion matrix, the best suitable topology was formed by 6 input nodes (which are the non-functional requirements of QoS as well as the system state indicators represented by the CPU load, memory and storage usage), two hidden layers, with 7 and 12 nodes respectively and the hyperbolic tangent as an activation function, and 4 output nodes representing the predicted values. The used learning rate was $1e^{-3}$. With this configuration the Mean Squared Error (MSE) was 0.0188 (RMSE=0.1371) after 500 epochs (Figure 2).

To construct and train the MANFIS, we employed the same input/output data set that had been used previously in MLP with 3500 values, which we believe represents a Cloud server working in different situations. The network was also constructed by means of Matlab software. As expected, MANFIS does not need a lot of manual configuration. This is due to the fact that the parameter configurations are adjusted by internal mathematical functions based on input data. However, the designer of the network still has to select the best method to partition the input space. In our case, due to the number of variables involved in the input, we cannot use grid partition as a method for input space partitioning (it generates rules by enumerating all possible combinations of membership functions of all the input and this leads to an exponential explosion of the rules). In view of this, we decided to use subtractive clustering which produces scattering partition. With this method only a limited number of computational resources are needed due to small number of rules in force. In this case, the only information required is the degree of influence of the cluster center on each input and output dimension, assuming the data falls within a unit hyperbox (range [0 1]). If a smaller cluster radius is specified, this will usually yield more smaller clusters in the data, and hence result in more rules.

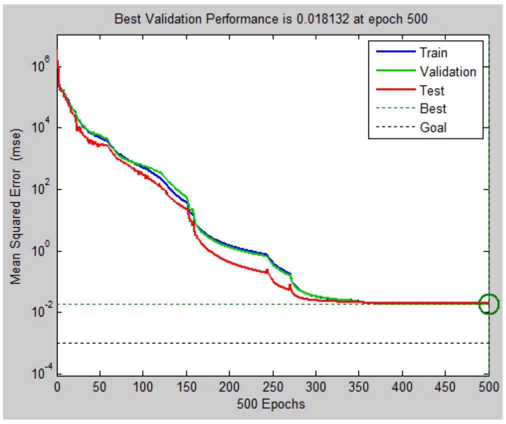

Figure 2: Mean Squared Error after 500 epochs

As in the case of MLP, by means of the 10-fold cross-validation method, the best suitable configuration was achieved with 0.15 for cluster radius to comply with SLA, memory and storage input and 0.5 for storage input. For comparative purposes, we used a training set which was done in 500 epochs too. With this configuration, the resulting RMSE was 0.0707 (Figure 3) in contrast with 0.1371 in MLP. The total number of rules generated by the space partitioning was around 15 for each input (in tests using grid partition with 3 membership functions for each of the 6 entries, the total was 3^6, which equals 729 rules). As expected, we found that if we reduce the cluster radius values, the number of rules and amount of computational time needed to calculate these rules increases dramatically. In this case, to be computationally efficient, we must reduce the training set and hence reduce the generalization power of the network.

Moreover, with regard to the low RMSE of MANFIS compared to MLP, if we analyze the convergence graphs of MLP and MANFIS we can note that the convergence of MANFIS occurs at a much faster rate than MLP in training mode.

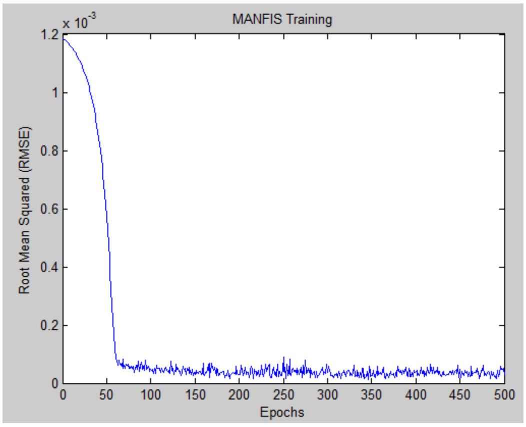

Figure 3. RMSE after 500 epochs

At training mode, we can conclude that MANFIS has a greater power of prediction due to a low RMSE. To evaluate and validate the predictive power of both approaches in our Cloud environment, we used the same scenario as that employed in our previous work [Schubert, F., Rolim, C. O. and C.B. Westphall 2011] which is depicted in Table 4. It contains input values and expected output (gray rows) by a prediction mechanism in different scenarios that simulates a normal server load and an overloaded server. Again we stress that we do not consider “high level” attributes related to business impact and economics relations like monetary costs.

Table 4. Test Scenarios

Attributes	*Scenario 1*	*Scenario 2*	*Scenario 3*
Concurrent Requests	85	287	958
Requests per Second	225	163	328
Time per request	62	193	230
Memory Usage	64.2	287	434
Storage Usage	85	287	958
System Load	0.8	3.8	10.4
Expected SLA	0	1	1
Expected Memory Provisioned	0	347.84	562.56
Expected Storage Provisioned	0	4383	5054
Expected Cores Provisioned	0	4	10

These scenarios have been generated on the basis of historical data obtained from the test environment for their inputs and the predicted output has been calculated by means of the following allocation functions:

$$Mal = 128MB + MpReq$$
$$Sal = 4096MB + SpReq$$
$$Cal = Reqs * 0.02$$

The *Mal* function represents the memory allocation that constitutes 128MB for the base instance plus the amount of memory needed for the average number of incoming requests. The *Sal* function stays for storage allocation, saying that the provision network will result in a 4096 Megabytes basic system storage plus the storage needed per incoming request. The *Cal* function handles the provision of CPU cores, where each request will be multiplied by 0.02. This value has been calculated on the basis of a systems analysis used for processing consumption per test request. The functions were defined on the basis of the test environment and test web application. For simulation purposes, the *MpReq* (Memory per Request) and *SpReq* (Storage per Request) variables have been considered as 1 Megabyte of RAM memory and disk storage respectively.

The values obtained from the test scenarios outlined above, were included as input in both networks (MLP and MANFIS) and the output can be seen in Table 5. If we compare the expected output values of Table 4 (gray rows - Expected SLA, Expected Memory Provisioned, Expected Storage Provisioned, Expected Cores Provisioned) with the predicted output of MLP and MANFIS (Table 5 – bold values), we can draw the following conclusions: The first test scenario depicts a normal server load with a number of requests that do not violate the SLA. The predicted output values of MLP and MANFIS were considered to be acceptable with a small precision error. This means that both networks are able to predict a normal state in the instance that does not violate the SLA. The service was not degraded and the neural networks were able to identify it without provisioning any resource.

A different result can be seen in Test Scenario 2. The SLA and Storage attributes have been correctly predicted by both networks. In MLP the number of CPU Cores has been rounded to low and the Memory predicted was 23% larger than expected. On the other hand, the number of CPU cores predicted by MANFIS was 15% larger than expected and the Memory provisioned was around 1% higher than expected.

In Test Scenario 3, both networks correctly predicted the SLA violation. However, MLP was wrong in its prediction of the number of Cores; the Memory was 27% lower than expected and the Storage provisioned was around 10% lower. The results of MANFIS were more accurate. The Memory, Storage and number of Cores predicted was around 1% higher than expected. These values can be considered to be correct.

Table 5. Results of provisioning in different scenarios*

	MLP Provisioning			*MANFIS Provisioning*		
Scenario / *Attributes*	*Scenario 1*	*Scenario 2*	*Scenario 3*	*Scenario 1*	*Scenario 2*	*Scenario 3*
SLA Status Predicted / Expected	**0** / 0	**1** / 1	**1** / 1	**0** / 0	**1.009** / 1	**1.00** / 1
Memory Predicted / Expected	**0.0002** / 0	**431.2512** / 347.84	**408.38** / 562.56	**0.0027** / 0	**345.30** / 347.84	**563.49** / 562.56
Storage Predicted / Expected	**0** / 0	**4383** / 4383	**4557** / 5054	**0.0018** / 0	**4383.7** / 4383	**5058.25** / 5054
Cores Predicted / Expected	**0** / 0	**3** / 4	**9** / 10	**0.0087** / 0	**6.12** / 4	**10.1732** / 10

**In bold are predicted values. The values after the slash came from table 4 and are shown to facilitate comparison*

The results show that MLP is a useful way of predicting values in a Cloud environment but more manual intervention is required to configure its parameters and a lot of run/test procedures to obtain satisfactory results. In contrast, MANFIS is a model that does not need a lot of manual intervention. We just had to make a few adjustments to the cluster radius parameters and the network itself did the rest. The prediction results of MANFIS were more accurate than MLP and it was clear that the training stage was faster and more computationally effective. Furthermore, in the training stage, we noted that MANFIS could achieve the desired RMSE with fewer values in the data training set. However this data must contain the maximum and minimum values in the universe of discourse, otherwise the generalization power of the network will be degraded.

To summarize, both types of network successfully detected the shortage of resources and predicted values to deal with these situations. MANFIS was found to have a better predictive capacity than MLP in our tests and required less effort for configuration and training. This leads us to believe that it is more suitable to provide resource provisioning for Cloud computing environments. Nevertheless, apart from its results, it is a useful approach and it would be worth confirming its effectiveness with more experiments in real environments.

6. Conclusion and Recommendations for Future Studies

The dynamic provisioning of resources remains a challenge for cloud computing. The use of neural networks for provisioning and SLA violation detection is one of the solutions proposed by UC Berkeley in its technical report on cloud [Armbrust, M. et al. 2009]. It applies where pervasive and aggressive use of machine learning as a tool for diagnosis, prediction and resource provisioning is used to enable dynamic scheduling, automatic reaction to performance problems and the automatic management of cloud systems [Armbrust, M. et al. 2009].

In this paper, we conducted an analysis of two type of neural network called MLP and MANFIS, a different hybrid type that blends learning capacities with fuzzy logic and provides good predictive capacities, as an underlying mechanism for cloud resource provisioning. The results show that MANFIS is more suitable for our needs. However to be used in a real environment, special precautions must be taken specially in the choice of the clustering method of input data since it can have an impact on the number of rules and the generalization power of the network.

It should be stressed that there is a lack of a mathematical approach to train the MANFIS ("on the fly") to improve the application of this network in different scenarios. Hence, we are seeking a way to do this in future studies another recommendation for future work is the development of a module that couples a neural model with Cloud simulators, such as *CloudSim* [Calheiros R., et al. 2009.]. This combination would allow the effectiveness of neural networks to be measured in environments where provisioning is closer to reality. Another topic of interest for future developments is an analysis of the economic feasibility and business impact from such techniques in enterprise or research facilities.

Finally, we conclude that the future of cloud computing might derive some benefit from statistical learning machines, with neural networks playing an important role in the forecasting, prediction and evaluation of cloud computing environments.

References

Abraham, A. (2005) "Adaptation of Fuzzy Inference System Using Neural Learning,". In: Springer-Verlag Berlin Heidelberg.

Aib, I. and Daheb, B. (2007), "Management, Control and Evolution of IP Networks," In: ISTE 2007. Chapter: SLA Driven Network Management.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson , D., Rabkin, A., Stoica, I. and Zaharia, M., (2009), "Above the Clouds: A Berkeley View of Cloud Computing," University of California at Berkeley, Tech. Rep.

Balen, D. and Westphall, C. B. (2011) "Experimental Assessment of Routing for Grid and Cloud," In: International Conference on Networking – ICN 2011.

Benmiloud, T. (2010) "Multioutput adaptive neuro-fuzzy inference system". In: Proceedings of the 11th WSEAS international conference on Neural networks and 11th WSEAS international conference on evolutionary computing and 11th WSEAS international conference on Fuzzy systems (NN'10/EC'10/FS'10), World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA.

Buyya, R., Pandley, S. and Vecchiola, C., (2009), “Cloudbus Toolkit for Market-Oriented Cloud Computing,” Proceeding of the 1st International Conference on Cloud Computing (CloudCom 2009, Springer, Germany), Beijing, China.

Calheiros , R. et al. (2009). “CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services”. In: University of Melbourne, GRIDS Laboratory.

Calheiros R. , et al. (2009). “CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services”. University of Melbourne, GRIDS Laboratory, 2009.

Elabd, S. and Schlenkhoff A. (2009), "ANFIS and BP neural network for travel time prediction". In: World Academy of Science, Engineering and Technology, 57, pp 116–121.

Gomathi, V., Ramar, K. and Jeevakumar, A. S. (2009), "Human Facial Expression Recognition using MANFIS Model", In: Proceedings of World Academy of Science Engineering and Technology, 38, pp 338-342.

Grimes, J., Jaeger, P. and Lin, J. (2008), “Weathering the Storm: The Policy Implications of Cloud Computing,” unpublished.

Jang, J. R. (1993) “ANFIS: Adaptive-Network-Based Fuzzy Inference System,” In: IEEE Transactions on Systems, Man, and Cybernetics, vol 23, pages 65—685.

Mackey, M.C. and Glass, L. (1977) “Oscilation and Chaos”, in Physiological Control Systems, Science, Vol 197, pp.287-289.

Mallick, S. and Hains,G. (2011), “Virtualization based cloud capacity prediction”, In: International Conference on High Performance Computing and Simulation HPCS 2011, IEEE, pp 849–852.

Marza, V. and Teshnehlab, M. (2009) “Estimating Development Time and Effort of Software Projects by using a Neuro_Fuzzy Approach,” In: Advanced Technologies, Kankesu Jayanthakumaran (Ed.), ISBN: 978-953-307-009-4, InTech.

Pramanika, N. and Kumar, R., (2009) "Application of neural network and adaptive neuro-fuzzy inference systems for river flow prediction", In: Hydrological Sciences Journal,Taylor & Francis, pp 247-260.

Schubert, F., Rolim, C. O. and C.B. Westphall (2011), “Aplicação de Algoritmos de Provisionamento Baseados em Contratos de Nível de Serviço para Computação em Nuvem”, In: XXiX Simposio Brasileiro de Redes de Computadores - IX Workshop em Clouds, Grids e Aplicações (WCGA 11), Campo Grande – MS.

Yilmaz, I. and Kaynar, O. (2011), "Multiple regression, ANN (RBF, MLP) and ANFIS models for prediction of swell potential of clayey soils", In: Expert Syst. Appl. 38, 5, 5958-5966.

Zhang, J., Chung, H.S. and Lo, W. (2008). "Chaotic Time Series Prediction Using a Neuro-Fuzzy System with Time-Delay Coordinates", In: IEEE Transactions on Knowledge and Data Engineering, 20(7), 956-964.

Zohar, E. and Cidon, I. (2011) “The Power of Prediction : Cloud Bandwidth and Cost Reduction”, In: Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM (SIGCOMM '11).