An Energy-Efficient Self-Provisioning Approach for Cloud Resources Management

Hanen Chihi
University of Tunis ElManar
Institut Superieur d'Informatique/SOIE
Tunis, Tunisia
hanen.chihi@gmail.com

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

Khaled Ghedira
University of Tunis/SOIE
Tunis, Tunisia
Khaled.Ghedira@isg.rnu.tn

ABSTRACT

In recent years, energy conservation has become a major issue in information technology. 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. Another unique feature of our work is a resources administration strategy for energy saving in the cloud. Such a strategy is implemented as a self-administration module. We show that the proposed approach gives promising results.

Categories and Subject Descriptors

Information Systems Applications [**Artificial Intelligence**]: Software Engineering

General Terms

Theory

Keywords

Cloud computing, prediction, recurrent self-organizing map, autonomic computing

1. INTRODUCTION

The issue of energy consumption in distributed computing systems raises various monetary, performance and environmental concerns. From a financial and environmental point of view, the reduction of power consumption is important, but these reforms must not lead to a degradation of the performance of computer systems. These conflicting pressures create a series of complex problems that must be solved in order to complete the greening distributed computing systems.

Cloud computing appears as a promising model to serve the increasing demands of today's complex applications for computing power [1] [15]. Virtualization is generally promoted by research work as a means to decrease the energy consumption of large-scale distributed systems [9]. However, the studies often lack real data on the electric consumption of virtualized infrastructures.

By enabling the design and development of systems that can adapt themselves to meet requirements of performance, fault tolerance, reliability, security, etc., without manual intervention, autonomic computing is a suitable candidate to endow the cloud with self-management capabilities (usually known as self-*properties) which are instantiated by self-configuration, self-optimization, self-healing and self-protection [10]. Other instantiations of self-* properties have also been adopted. For example, autonomy of maintenance is proposed by Chainbi [4] and system adaptation and complexity hiding is proposed by Tianfield and Unland [18]. Every element in an autonomic system consists of two main modules: the managed element that performs the required services and functionality, and the manager that monitors the state and context of the element, analyzes its current requirements and adapts it to satisfy the needs.

In this study, we develop a green auto-configuration strategy to ensure the auto-scaling of virtual machines (VM) within a cloud. The auto-scaling of cloud resources reduces power consumption by adapting the computational resources according to the actual workload. Auto-scaling can be ensured by a scaling-up and scaling-down. The scaling-up consists in adding computational resources. The scaling-down consists in removing unused resources.

There has been a great deal of research on dynamic resource prediction and allocation for physical and VM and data-centers of VM [13] [17]. Moreover, there were many studies devoted to autonomic configuration of VM parameters. For example, in [11], 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 [12][6] which are commonly suggested, but tend to scale very poorly with the number of resources.

Actually, it is hard to implement cloud auto-scaling methods that aim to reduce energy consumption and environmental impact [14]. Auto-scaling must ensure that resources can be provisioned quickly to meet users' requirements workload. 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 Networks (NN) to predict in advance needed resources regarding users' demands. So predicted resources can be booted and configured in order to be allocated rapidly.

As Cloud computing is still an emerging paradigm for distributed computing, there is a lack of defined standards, tools and methods that can efficiently tackle the infrastructure and application level complexities. In the present work, we propose a module that couples a neural model with the cloud simulator CloudSim [3]. This combination would allow the effectiveness of NN to be measured in environments where provisioning is closer to reality. By extending the basic functionalities already exposed with CloudSim, we will be able to perform tests based on specific scenarios.

The proposed model describes application's workloads including information of cloud users, number of users and data-centers, and number of resources in each data-center. Then, it generates information about requests' response time, requests' processing time, and other metrics. By using this information, developers can determine the best strategy for resources' allocation, strategies for selecting resources to serve specific requests, and costs related to such operations.

The proposed approach is based on the cloud Infrastructure as a Service model (IaaS). In fact, cloud customers have access to the Cloud resources by running services. The objective of this work is to allow resources providers to optimize the number of active resources and minimize energy consumption. The novelty of our solution lies in its integration of a neural predictor and the development of policies for dynamic training and dynamic provisioning of additional servers, as well as considering server's timing requirements. The neural network uses the mean squared error (MSE) to evaluate and update the neural network's weights. In the proposed approach, we added an evaluation measure that takes into account the optimization of consumed energy.

Another unique feature of our work is the design of a resources self-administration module that mainly performs the following operations: resources allocation, resources release, resources configuration, monitoring of the resources status and the resources pool status.

The remainder of this paper is organized as follows. Section 2 presents the recurrent self-organizing map. Section 3 describes the problem and the system model. Experimental results and a case study are presented in the section 4. Section 5 deals with related work.

2. RECURRENT SELF-ORGANIZING MAP

The self-organizing map, introduced by Kohonen, is an artificial neural network based on an unsupervised competitive learning [20]. We concentrate on the SOM network known as a Kohonen network 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. Fig. 1 shows a bidimensional SOM.

In our approach, we are interested in the use of recurrent neural networks (RNN). They employ feedback connections and have the potential to represent certain computational structures in a more parsimonious fashion. We argue that that the instantiation of rules from predictors not only gives the user confidence in the prediction model, but also facilitates the use of the model and can lead to a deeper understanding of the process.

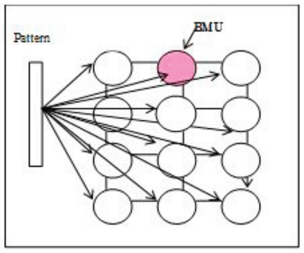

Figure 1: Bidimentional SOM.

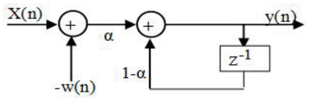

Figure 2: Learning process of RSOM.

We present as an extension to the Self-Organizing Map, the Recurrent Self-Organizing Map (RSOM) [20] that allows storing certain information from the past input vectors. 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 (Fig. 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 $\boldsymbol{y}_i(n)$ in each neuron of the map is updated as follows:

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

where is the weight vector, *x(n)* is the input pattern, $\mathbf{0} < \alpha < 1$ is the memory coefficient and is the leaked difference vector for the neuron i at step n while large $\boldsymbol{\alpha}$ 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

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

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

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

where h_{bmu_i} is the value of the neighbourhood function.

3. RESOURCE SELF-MANAGEMENT BASED ON AN RECURRENT NEURAL NETWORK

This section presents a preliminary architecture towards autonomic cloud, focusing primarily on its components and their main requirements.

3.1 Provisioning scenario

We describe here the proposed self-adaptable provisioning architecture [5]. Dynamic provisioning refers to the ability of dynamically acquiring new resources and integrating them into the existing infrastructure and software system. As shown in Fig. 3, the proposed architecture is based on the standard cloud services architecture. The key feature of this architecture is the separation of mechanisms and policies. As mentioned above, the important feature of self-adaptation to enable autonomic computing is the self-control capability, which can regulate the autonomic systems by interpreting the autonomic model and policies and injecting the new policies into the system.

The dynamic provisioning infrastructure is mainly the result of the collaboration between two services: the resource self-administration module and the prediction module. They are in charge of making the provisioning happen by interacting with IaaS providers and local virtualization facilities. The proposed model provides the following functionalities:

- Requesting VM from a set of IaaS providers;
- Detecting dynamically the need of additional resources;
- Controlling the lease of resources and eliminating resources that would otherwise remain idle and waste power using a neural predictor;
- Regulating the autonomic systems by interpreting the autonomic model.

The system is completely modular and can be fully customized to activate and control the behaviour of dynamic provisioning. The proposed model integrates a autonomic model and policies that is endowed by an autonomic provisioning model, a knowledge data base, a self-control policy and users' requirement analysis module.

The autonomic provisioning is the result of the collaboration of several components including the resource pool monitor, the resource self-administration module, and the resource predictor [5].

3.2 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:** 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:** 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 $t + \lambda$. [20].
- **Effector:** 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 time $t+\lambda$, RSOM must predict the number of potential applications 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.

3.3 Resources self-administration module

The proposed self-administration module provides abstraction with complete customization of cloud resources. Fig. 4 presents the process of the self-administration operation. The most common administrative operations in IaaS are:

- **Allocation of resources:** It attributes to the user portion of exploitable resources.
- **Deployment:** Deployment in the IaaS systems mainly concerns VM file.
- **Configuration and Startup:** The configuration operation depends both on business demands and also the policy of allocating. The configuration includs the amount of memory, the number of processors, the locus of execution of the VM and the VM file.
- **Reconfiguration:** Reconfiguration operations occur during the execution of the IaaS and alter its composition.

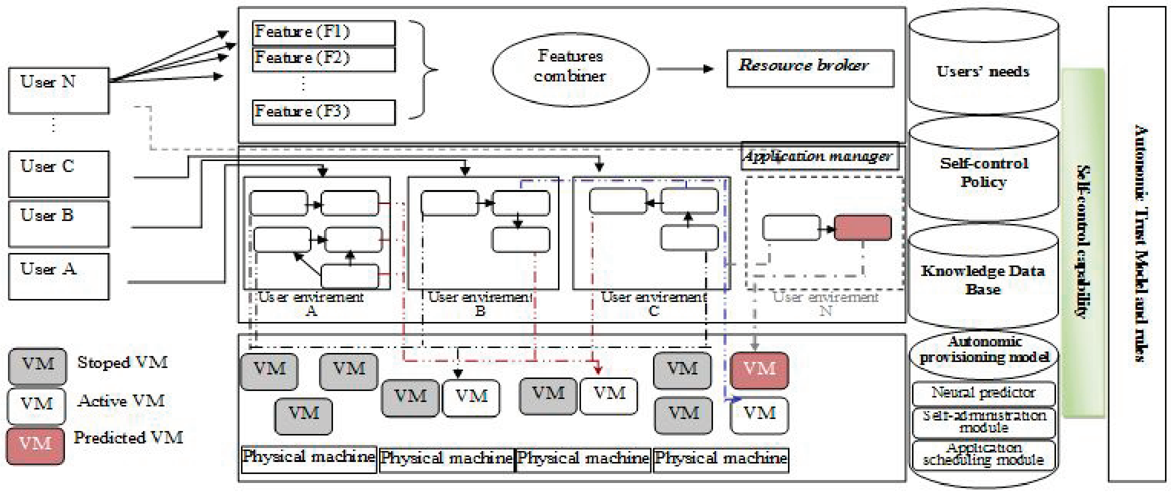

Figure 3: The Conceptual Architecture of the Autonomic cloud Computing for resources provisioning.

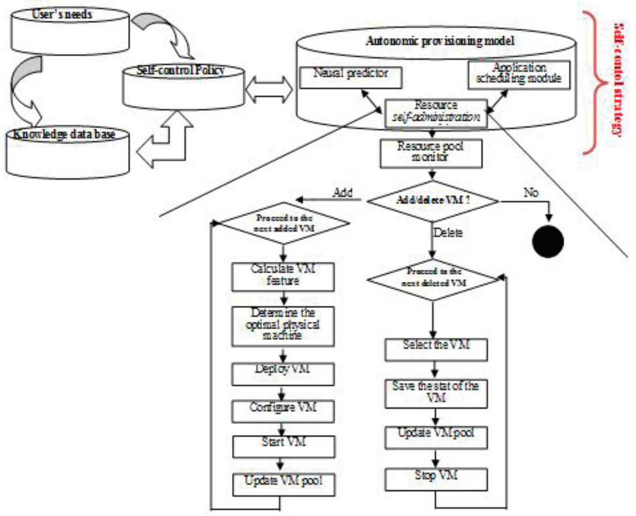

Figure 4: Structure of the resource self-administration module.

Self-control strategy has been used to model the global and local feedback control loops in the autonomic provisioning model while maintaining the cloud trustworthiness. The local control loop in an autonomic entity typically follows four steps: monitoring through sensors, analyzing and planning according to knowledge bases and policies, then executing through actuators. The local control loop is restricted to handle internal reactions and external interactions with associated managed elements and environments. As shown in Fig. 4, the global control loop involves self-provisioning model, users' needs, knowledge data base, and self-control policies. The self-provisioning model collaboratively regulate various behaviors using self-control functionalities and knowledge.

3.4 Energy model

The energy consumption of physical machine results of the computer equipment. In our work, we take into account the energy consumption due to computing (computer equipment).

In our description, clients shall pay a sum of money to run their services. This award is the result of the power consumption of each of cloud's physical machines in the calculation. Based on this cost, the cloud provider sets a price for the sale of service for customers.

Energy consumption by Cloud nodes is mostly determined by the CPU, memory, disk storage and network interfaces. Moreover, the CPU utilization is typically proportional to the users' workload. This fact justifies the technique of switching idle servers to the sleep mode to reduce the total power consumption. Therefore, in this work we use the power model defined in (eq. 5).

The overall energy consumption of a cloud computing system, an environment with n nodes and a centralized storage device, can be expressed as the following formula:

$$E_{Cloud} = nxE_{node} + E_{Storage} \quad . \qquad (4)$$

E_{node} represents the energy consumption of the physical computing node. $E_{Storage}$ represents the energy consumption of the storage device. The power consumption of the cloud computing component (CPU or Storage) could be expressed as follows:

$$P_i = k \times P_{max} + (1-k)P_{max} \times e \quad . \qquad (5)$$

P_{max} is the maximum power consumed when the server is fully utilized; k is the fraction of power consumed by the idle server; and e is the CPU utilization.

The Cloud node changes over time due to the workload variability. Thus, the node utilization is a function of time and is represented as e(t). Therefore, the total energy consumption by a component i can be defined as an integral of the function of the power consumption over a period of time as shown in (eq. 6).

$$E_i = \int_{t1}^{t2} P(e(t))dt \quad . \qquad (6)$$

The sensor component of the self-provisioning model collects information about energy consumption. This information is used to regulate the training of the neural predictor.

4. 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 [19]. In order to compare different methods, we have examined the following 3 different neural training modes:

- Supervised static training: multi-layer perceptron (MLP).
- Unsupervised static training: SOM network.
- Unsupervised dynamic training: RSOM network.

4.1 Creating the Workload Request Scenario

The CloudSim simulation layer provides support for modelling and simulating cloud environments including dedicated management interfaces to handle VM, memory, storage and bandwidth [3]. The simulator consists of a set of fundamentals components that are: Datacenter, CloudBroker, Host, Vm, Cloudlet, . . .

In order to allow simulation of dynamic behaviors within CloudSim, we have made a number of extensions to the existing framework.

The model creates user and broker components dynamically and it manages, at runtime and regarding users' request variation, "Datacenter", "Host" and "VM" components. After creation, new entities automatically authenticate themselves in cloud to allow the discovery of dynamic resources.

To evaluate our prototype we ported the "Rain Workload Generator" driver to communicate with CloudSim components such as "Datacentres" and "CloudBroker". We create a "RequestGenerator" and then drive load against a small data-center running in the CloudSim framework. Our goal is to study the performance of the cloud infrastructure using Rain and the proposed extension of the CloudSim simulator.

The "Rain Workload Generator" harmonizes an entire workload experiment. It loads the "ConcreteScenario", initializes the "Scoreboard" where the results are collected and summarized a run the "ConcreteGenerator".

The "ConcreteScenario" is the component that triggers the workload generation scenario. It contains the parameters of an experiment such as the maximum number of users to emulate, the duration of an experiment and sequence of request to adopt during a run.

The "ConcreteGenerator" creates demands for a single active user. It is initialized with a scenario and determines the next query on the database of the previous application and all other internal criteria.

The predictor module triggers a method that searches for cloud historical data and calculates future states of the cloud infrastructure. Then, it invokes a method from a self-administration module that supervises the configuration and the reconfiguration of the cloud infrastructure.

The "CloudBroker" queries the "CISRegistry" component for a list of actives "Datacenters" which offer services matching the user's application requirements. Then, the "CloudBroker" deploys the application in the selected data-center.

4.2 Data analysis

The development of an application in neural networks should be undertaken in the definition of input and output data. The input data is provided by the QoS attributes and system state variables used in cloud environment:

- Requests: the number of requests per second;
- Time per request: runtime per second to complete each 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 processes are kept in wait or execution state.
- Storage Consumption: storage size in Megabytes consumed by the requests.

In the output, the network predicts the resource provisioning needed by the environment, using previous context. These values are: processor cores number, amount of memory, amount of storage.

4.3 Setting of neural networks

To train the different neural networks, 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 generalization power of 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 [20]. 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 λ 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 (MLP, SOM or RSOM). Therefore, we achieve the chosen of parameters of prediction model based on each neural network through the program. In addition, we determine the optimal parameters by comparing the MSE values given by different neural networks 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 neural network prediction 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.

To evaluate and validate the predictive power of the approaches in our cloud environment which is depicted in Table 1. It contains input values and expected output by a prediction mechanism in different scenarios.

$$Store_{all} = Store_{BS} + Store_{Req}. \tag{7}$$

$$Mem_{all} = Mem_{BI} + Mem_{Req}. \tag{8}$$

$$CPU_{all} = NB_{Reqs} x CPU_{Req}. \tag{9}$$

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

The $Store_{all}$ represents the storage allocation that constitutes the sum of $Store_{BS}$, the basic system storage, and $Store_{Req}$ (Storage per Request), the storage needed per incoming request. The Mem_{all} function represents the memory allocation that constitutes Mem_{BI} for the base instance plus the amount of memory needed for the number of incoming requests Mem_{req} (Memory per Request).

The CPU_{all} function handles the provision of CPU cores, where each request will be multiplied by CPU_{req} (processing time of a request). For simulation purposes, the Mem_{req} and $Store_{req}$ variables have been considered as 2 Megabyte of RAM memory and disk storage respectively. CPU_{req}, Mem_{BI}, $Store_{BS}$ have been fixed, respectively, to 0,05, 128 Megabyte and 4 Gigabyte.

4.4 Evaluation analysis

The values obtained from the test scenarios outlined above, were included as input in used neural network (MLP, SOM and RSOM) and the output can be seen in Table 2. From the predicted output of MLP, SOM and RSOM (Table 2), 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, SOM and RSOM were considered to be acceptable with a small precision error.

A different result can be seen in the second test scenario. The CPU cores and storage attributes have been predicted by studied neural network. In MLP the number of CPU cores has been rounded to low and the memory predicted was 20% larger than expected. On the other hand, the number of CPU cores predicted by SOM was 17% larger than expected and the memory provisioned was around 1% higher than expected.

Regarding the RSOM, memory, storage and the number of processor cores have been predicted with a small error of about 0.15, 0.009 and 0.09, respectively.

In the third test scenario, used network correctly predicts the SLA violation. However, MLP was wrong in its prediction of the number of cores; the memory was 25% lower than expected and the Storage provisioned was around 9% lower. The results of SOM were more accurate. The memory, storage and number of cores predicted was around 1% higher than expected. Regarding the RSOM, the predicted values can be considered to be correct.

Table 1: Test scenarios

	Test 1	Test 2	Test 3
Requests per second	250	300	350
Time per request (s)	81	200	300
Memory usage (Mb)	64	297	474
Storage usage (Gb)	100	300	1000
CPU cores usage	12	15	17
Concurrent requests	100	293	1000
Expected SLA	0	1	1
Estimated memory (Mb)	120	384,91	581,56
Estimated storage (Mb)	2048	4423	5254
Estimated Cores	2	5	12
Estimated VM number	1	3	6
Estimated Data-centers number	1	1	1

According to information provided by the neural predictor endowed by the MLP, SOM or RSOM, we deduced the number of VMs and datacenters (the last two lines of Table 2) necessary to respond to user requests.

Only the predictor using the RSOM could deduce the appropriate number of VMs for the three test scenarios. However, the MLP gave values lower by one unit and the SOM has provided for an additional unit for the second test.

The results show that MLP and SOM 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.

Regarding the RSOM, it is a model that does not need a lot of manual intervention; the network itself did the rest. The prediction results of RSOM were more accurate than MLP and SOM and it was clear that the training stage was faster than MLP neural network and more computationally effective.

The simulation results are presented in Table 2 show that energy consumption can be significantly reduced by using RSOM predictor.

To summarize, used types of neural network successfully detected the shortage of resources and predicted values to deal with these situations. RSOM 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.

5. RELATED WORK

Table 2: Provisioning results of different scenarios

		MLP			SOM			RSOM	
	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3
SLA status	0,01	1	1	0,001	0,01	1	0	1	1
Memory (Mb)	121,3	361,89	436,17	120,98	385,30	582,49	119,02	384,99	581,62
Storage (MB)	2101	4424	4781	2050,9	4423,7	5258	2049,99	4428	5254,10
CPU cores	2	4	11	2	5,85	12,32	2	5,09	12
Number of VMs	1	2	5	1	3	7	1	3	6
Number of datacenters	1	1	1	1	1	1	1	1	1
Consumed Energy W*sec	120	3258,05	5391,1	131	5248,03	5727,12	99	2989,12	5248,04

Provisioning of cloud resources has been a topic of research interest and development for many years. To address the growing challenge, techniques from many disciplines, such as artificial intelligence, machine learning, neural science, and others, where integrated synergistically. In what follows, we present the state of the art of cloud resources provisioning models integrating the properties of autonomic computing, and using artificial intelligence.

There has been a great deal of research on dynamic resource allocation for physical machines and VM and datacenters. Van et al. [11] propose an autonomic resource management approach based on two-level architecture. In the local loop, authors resort to utility functions to map the current state of each application to a scalar value that quantifies the ”satisfaction” of each application with regard to its performance goals.

Foster et al. [8] address the problem of deploying a datacenter of virtual machines with given resource configurations across a set of physical machines. Czajkowski et al. [6] define a Java API permitting developers to monitor and manage a data-center of Java VMs and to define resource allocation policies for such clusters.

Bolik et al. [2] present a machine learning based statistical approach in order to calculate the performance of cloud infrastructure and to determine the optimal number of resources required to maintain the performance of cloud applications.

Duy et al. [7] 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 is a multi-layer perceptron neural used to calculate future load demand based on historical demand. The ON/OFF algorithm dynamically adjusts resources allocations to minimize the number of resources running.

Shi et al. [16] 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.

Kliazovich et al. [14] proposed a GreenCloud. It is a cloud simulation toolkit that provides supports for modeling of cloud-based virtual resources for energy-aware cloud computing. According to the workload distribution, the simulator is designed to capture details of the consumed energy by cloud resources. The GreenCloud offers users a detailed grained modeling of the energy consumed by the components of data-center. Moreover, it offers a thorough investigation of workload distributions.

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 [6] [8], as well as statistical machine learning approach to predict in [2]. 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 [7]. All of them did not answer whether neural networks and time series analysis are fit for predicting different kinds of resources provisioning.

Both models in [7] and [16] offer neural predictors: the first one is based on a Back propagated (BP) predictor and the second one is based on a RBF network.

Described networks don’t contain internal dynamics. In order to use these networks for the identification of nonlinear dynamic systems external dynamic elements are needed. Networks having internal dynamics like dynamic multi-layer perceptron and recurrent networks have also been tried for prediction purposes. This is an important reason why we choose the 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.

By studying the different 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.

6. CONCLUSION

Computing systems including hardware, software, communication and networks are growing towards an ever increasing scale and heterogeneity, becoming overly complex. To cope with the growing complexity, autonomic computing focuses on self-adaptable computing systems that exhibit self-* properties to the maximum extent possible without human intervention or guidance. In this work, we have proposed an autonomic green provisioning approach for energy saving in the cloud. 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 envision to adopt an agent-based architecture for cloud infrastructure self-administration. A related study is underway. We are also developing a self-optimization scheduling application in the cloud.

7. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A.Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of ACM*, 53(4):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*, 2010.

[3] R. Calheiros, R. Ranjan, A. Beloglazov, C. D. Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw., Pract. Exper.*, 41(1):23–50, 2010.

[4] W. Chainbi. Agent technology for autonomic computing. *J. Transactions on Systems Science and Applications*, 1(3):238 – 249, 2006.

[5] H. Chihi, W. Chainbi, and K. Ghedira. Unsupervised neural predictor to auto-administrate the cloud infrastructure. *IEEE/ACM 5th International Conference on Utility and Cloud Computing, 3rd International Workshop on Green and Cloud Management, November 5-8, 2012. Chicago, Illinois, USA.*

[6] G. Czajkowski, M. Wegiel, L. Daynes, K. Palacz, M. Jordan, G. Skinner, and C. Bryce. Resource management for clusters of virtual machines. *Cardiff, UK*, pages 382–389, 2010.

[7] T. Duy, Y. Sato, and Y. Inoguchi. A prediction-based green scheduler for datacenters in clouds. *IEICE TRANS. INF. et SYST.*, 94(9):1731–1741, 2011.

[8] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang. Virtual clusters for grid communities, cluster computing and the grid. *CCGRID 06. Sixth IEEE International Symposium, Singapore*, pages 513–520, 2006.

[9] B. Golden. Virtualization for dummies. 1(3):70–90, 2007.

[10] I. Group. An architectural blueprint for autonomic computing. *http://www-03.ibm.com/autonomic/pdfs/AC*, 2003.

[11] V. H.N., F. Tran, and J. Menaud. Autonomic virtual resource management for service hosting platforms. *Workshop on Software Engineering Challenges in Cloud Computing, Vancouver, Canada : Canada*, pages 1–8, 2009.

[12] W. Iqbal, M. Dailey, D. Carrerab, and P. Janecek. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27.

[13] S. Kamboj, T. Estrada, M. Taufer, and K. Decker. Applying organizational self-design to a real-world volunteer computing system. *AAMAS workshop on Agent Design: Advancing from Practice to Theory*, 2009.

[14] D. Kliazovich, P. Bouvry, and S. U. Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *Global Telecommunications Conference (GLOBECOM)*, pages 1–5, 2010.

[15] P. Mell and T. Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6):1–7, 2011.

[16] 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*, 3(11):67–78, 2011.

[17] 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)*, pages 631–636, 2010.

[18] H. Tianfield and R. Unland. Towards autonomic computing systems. *Engineering Applications of Artificial Intelligence archive Vol. 17 Issue 7*, pages 689–699, 2004.

[19] Traces. Traces in the internet traffic archive. *http://ita.ee.lbl.gov/html/ traces.html*, June 2012.

[20] 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, Springer Verlag*, pages 421–426, 1997.