Proc. of the 2011 6th International Conference on System of Systems Engineering, Albuquerque, New Mexico, USA - June 27-30, 2011

# Prediction of Cloud Data Center Networks Loads Using Stochastic and Neural Models

John J. Prevost, KranthiManoj Nagothu, Brian Kelley and Mo Jamshidi
Electrical and Computer Engineering, The University of Texas at San Antonio, TX, USA

*Abstract*— **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 introduce a novel framework combining load demand prediction and stochastic state transition models. We claim that our model will lead to optimal cloud resource allocation by minimizing energy consumed while maintaining required performance levels. We characterize the ability of neural network and auto-regressive linear prediction algorithms to forecast loads in cloud data center applications. In this paper, the performance of our models against two sets of data at multiple look-ahead times is also presented.**

***Keywords; cloud, green computing, linear prediction, load forcasting, neural networks, optimization***

## I. Introduction

Cloud computing platforms are being increasingly utilized by industry, government and academia due to their ability to deliver robust, resilient and scalable computational power. In cloud computing data centers, Giga-bit speed, or faster, networks interconnect both physical and virtual computers. These systems are dynamically provisioned based on a determination of the required computing resources requested by the end user of the cloud application. This enables the cloud data center to remotely host applications and operations, rather than hosting individual applications in local on-site clusters or individual computers. However, the amount of energy consumed and therefore the operational costs of these datacenters are considerably high. The direct power consumed by the cloud resources is not the only problem. The aggregate wattage output of the cloud resources creates a secondary power cost due to the HVAC systems keeping the temperature of the datacenter within required operational levels. A recent report stated that 61 billion kilowatt-hours of power was used for datacenters and servers in 2006. This is equivalent to 1.5 percent of all US electricity usage during the same year. The power used by cloud datacenters is projected to double by 2011 [1].

It is understood that the total power consumed by the datacenter is proportional to the number of servers that are active at any point in time. The more active servers there are in the cloud, the higher the current power consumption. It is well known that the incoming requests for computational resources are not a constant. A system that allocates the minimum number of active servers to meet the current demand would therefore consume less power than a system that fails to take this into account.

Currently, there exist multiple cloud tools (i.e. RightScale Automation, Enomaly, Intridea's Scalar, etc.) that manage deployments and adapt to dynamic variation of the load demand. In addition, these tools respond to system failure as well as other custom-defined events. The RightScale Automation Engine both executes and manages deployments of the cloud services onto specific servers. As demand changes, servers can be added or decommissioned. As components fail, existing servers can adopt the failed system's roles or the system can decide to deploy new servers. When queues fill or empty, these tools can cause the grids to expand or contract automatically. Most of these systems allocate a maximum number of resources in a manner that attempts to ensure that all the Service Level Agreements (SLA) are satisfied. Resource allocation is minimally invariant to the dynamic load request. For this reason, the traditional deployment models fails to use the minimum number of resources to service submitted processing request for the specific goal of minimizing energy consumption. Our objective of the model is not limited to maintaining SLA, dynamically commissioning and decommissioning of the resources based on load. We model the system by predicting the load and jointly allocating tasks to the optimal number of processors and dynamically change states of the reserve processors to obtain minimum power state.

This paper attempts to impact this problem by looking at the specific characteristics of the ability to predict future loads. The relationship between the ability to predict the future loads and accuracy of the load prediction must be analyzed in order to optimally allocate the cloud resources. With an accurate model for the ability to predict future loads, the optimal system state (sleep/hibernate/active) of each of the cloud resources can be ascertained.

We examine auto-regressive linear prediction and neural network prediction methods to forecast the future load demand profiles. As a result of this examination, the relationship between prediction accuracy and the forecast length will be determined. The results ultimately provide a robust model for determining the correct amount of system resources required. A goal is to ensure that incoming requests are being properly serviced with a minimum amount of required power.

There are several areas where energy efficiency can be achieved in the operation of data centers. Server innovation, virtualization, high efficiency power supplies and optimal load management are a few examples. Prior research on the Dynamic configuration of web server clusters, with on/off capabilities, is proposed in [2-3]. The author's in [2] present a solution to the problem of determining the number of nodes that must be turned on in order to handle the load imposed by the system. The solution also includes turning off the lightly loaded systems by consolidating the load on other systems. The author's main thrust is to develop load balancing and unbalancing algorithms. These algorithms take into account the total load imposed on the cluster. Power and performance optimization is achieved by turning nodes off. However, the algorithm's decision to identify the number of nodes to add or remove requires the ability to predict the performance and power consumption of each node. These results imply that when the cloud is configured to dynamically change the state of the cloud resources, the overall performance will decrease relative to a cloud that is managing the resources statically. However, the solution is not optimal since the authors do not take into account the time required to change the server state from OFF to ON. Dynamic configuration of web servers was also presented in[4]. The authors assumed the probability distribution of the waiting time through queuing theory. The problem in this approach is that the assumption of M/M/1 queues is far from reality, as well as the fact that the queuing equations make the problem nonlinear and difficult to solve optimally at runtime.

There are many predictive policies developed, even for interactive terminals [5-6] that force the transition of servers to a low power state as soon as a component becomes idle. These methods wait for a certain time then gracefully reduce the running resources. There are a few common approaches to predict network loads such as linear prediction, neural networks and fuzzy logic. The distinction in these prediction approaches is that the models are based on either stochastic or discrete modeling. Stochastic models use distributions to describe the times between arrivals of user requests (inter-arrival times), the length of time it takes for a device to service a user request, and the time it takes for the device to transition between its power states ( i.e. sleep, idle and active). In discrete models, the transition state times are fixed. However, assuming that the time it takes to change from one state to another is fixed may not true in all scenarios. For example, the equipment may be degrading over time and therefore the transition times may vary. A green scheduling algorithm [6] integrated with neural network has also been proposed for optimizing resources in cloud. Neural network prediction estimates the dynamic incoming load, and serves as input to a green scheduling algorithm for turning servers on and off. These decisions work to minimize the number of currently running servers.

The stochastic characterization of the load prediction changes with the prediction interval. It is therefore not sufficient to simply build an accurate load predictor. The nature of how the forecast accuracy varies with prediction length must also be known. The nature of this relationship will be examined in this paper.

## II. Prediction Model Framework Applied to Cloud Data Centers

The architectural framework of the cloud resources and the respective controllers implied in this paper is shown in fig.1. It is proposed that cloud systems have the addition of a device responsible for predicting the incoming load as well as a device that examines the current performance profiles of all the available cloud resource nodes. The information gathered by these devices is fed into a controller that will be responsible for managing the states of the cloud resource nodes. This architectural framework is presented in fig. 1.

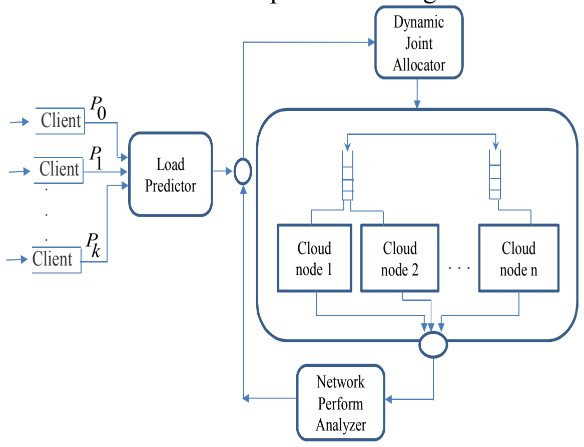

Fig. 1. Cloud architecture framework showing addition of prediction

## III. Prediction Algorithms

Two prediction algorithms are presented, Neural Networks and an Auto-Regressive Filter. A detailed explanation of each model is presented.

### A. Neural Network Model

Multi-layer neural network perceptrons have been applied to predict the future load of applications in cloud data centers. We predict the forecasted load by training the perceptrons in a supervised manner with a back-propagation algorithm. A back propagation algorithm consists of a forward pass and a backward pass. In the forward pass, the input vector is applied to the sensory nodes and its effect propagates throughout the layers and generates a set of outputs as a response. Moreover, in the forward pass the all the synapse weights are static. However, in the backward pass all the synapse weights are dynamically updated based on the error correction rule [8-10]. Dynamically updating the synapse weights forces the actual response to move toward the desired response, finally minimizing the error. A three- layer neural network predictor is illustrated in fig. 2.

The neural model receives three inputs x (n) from external information source and is applied to the input layer of the sensory nodes. Each neuron is composed of two units. The first unit adds products of weights coefficients and input signals. The second unit realizes a nonlinear function called the neuron activation function. The desired response vector d(n) is obtained at the output layer of the computation nodes. When the network is run, each layer performs the calculation on the input and transfers the result $O_c$ to the next layer [7].

$$o_c = h\left(\sum_{i=1}^{n} x_{c,i}\ W_{c,i} + b_c\right) \tag{1}$$

Where

$$h(x) = \begin{cases} \frac{1}{1+e^{-x}} & \text{if hidden layer node} \\ x & \text{if the output layer node} \end{cases} \tag{2}$$

Where Oc is the output of the current node, n is the number of nodes in the previous layer, $X_{ci}$ is the input of the current node from the previous layer, $b_c$ is the bias and $W_{ci}$ is the modified weight based on the error and it is calculated based on the equation 3.

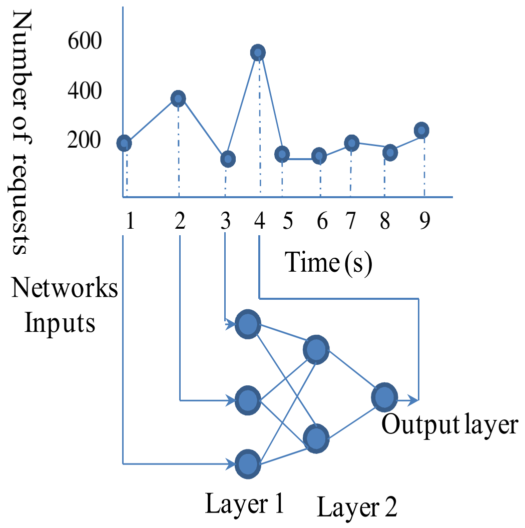

Fig. 2. Neural Network Model

$$\begin{pmatrix} \textit{Weight correction} \\ w_{c,i} \end{pmatrix} = (\textit{learning parameter}, \eta) * (\textit{local gradient}) * (\textit{input signal of the neuron}) \quad (3)$$

Computation of local gradient for each neuron of a multilayer perceptron requires knowledge of the derivative of the activation function associated with the neuron. H(x) is a sigmoid activation function for the hidden layers Activation function.

In order to make neural prediction effective, we need the neural network to be trained on a representative data set. Network training is an iterative process. In each iteration, the weight coefficients of the nodes are modified using new data from training data set. Each teaching step starts with forcing both input signals from training set. After this stage, we can determine values of the output signals for each neuron in each network layer.

Available data are often divided into two sets: training and testing. These sets can overlap and do not have to be continuous. The training set is a sequence that is shown to the neural network during the training phase. The network is adapted to it to achieve the required outputs (in other words, weights in the network are changed based on the outcome of this step). The last set, the testing set, is then used to test whether the network is able to work also on data that was not used in the previous process.

### B. *Task Prediction Using Auto-Regressive Prediction Filters*

We perform prediction using an autoregressive model with dithering of the observation sequence to smooth out numerical anomalies. For a prediction look-ahead of L samples, we model the observation data as the convolution of an impulsive source, $b(n)$, with the observation channel. The channel represents the number of accesses per discrete time index, $h(n)$, of the cloud data center. The dithering signal is labeled as $v(n)$ and has the purposeful effect of smoothing out the averaging. The prediction system is based upon an AR(P) Weiner filter. The training signal is $g(n) = h(n + L)$. The goal of the linear prediction filter is to observe the last N samples of the cloud channel and to update the predictor with coefficients reflecting the most recent statistical samples. The linear prediction coefficient are $w(n)$. The output of the prediction filter can be listed as

$$\hat{z}_n = \sum_{m=0}^{M_w-1} w(m) \underbrace{\left[ \sum_{r=0}^{L_H-1} h(r)b(n-m-r) + v(n-m) \right]}_{y(n-m)} \quad (4)$$

Thus the error signal, $\varepsilon(n)$, can be written as $\varepsilon(n) = \bar{g}_D^T \bar{b} - \bar{w}^T H \bar{b} - \bar{w}^T \bar{v}$ [see 11]. After much algebraic manipulation we can therefore solve for the linear prediction coefficients that minimize the mean square error term $\varepsilon^2(n)$ of the cloud data center observation as:

$$\bar{w} = \left(R_{vv} + H^H R_{bb} H\right)^{-1} H^H R'_{bb} \bar{g} \quad (5)$$

The system model for this equation is shown in fig.3.

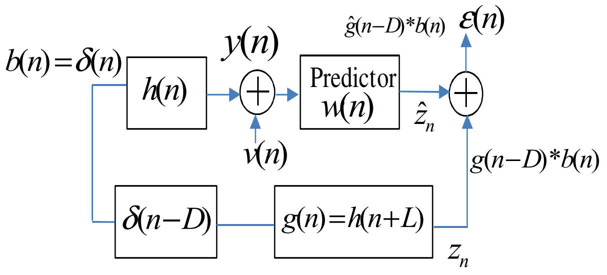

Fig. 3. System Model for Linear Prediction Filter

## IV. PREDICTION RESULTS

With the Linear Predictor and Neural Network models in place, sample network traffic data was used to train then simulate the forecasting of network loads for each of the two models.

The data used for the simulation was taken from [12]. The data represents URL resource requests of a WWW server at NASA and WWW server at EPA. The incoming requests were time stamped to the second. A summary of the data was created having the number of incoming requests per each sample period (1 second in this case).

### A. *Neural Network Simulation and Results*

For the neural network, the first one-thousand (1000) samples were separated from the data and used to train the network. Once the network was trained within a tolerable error, the rest of the data was fed in and both the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE) were calculated from the predicted results. This value was summed over the sample data then divided by the number of samples used in the simulation. The calculation of the MSE is shown below.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (d - \hat{d})^2 \quad (6)$$

$$d = \textit{Actual Load Value}$$
$$\hat{d} = \textit{Predicted Load Value}$$
$$N = \textit{Number of Samples}$$

The RMSE is the square root of the MSE.

This process was repeated with the same data, but with different prediction interval values, for a total of eight passes. The values for the look ahead intervals used in the simulations were (in seconds) 1s, 5s, 10s, 15s, 20s, 30s, 60s and 90s. Plots were obtained from each pass showing the Actual vs. Predicted Load Values for both load traces from NASA and EPA. Both plots are shown below in Fig.4a and fig.4b.

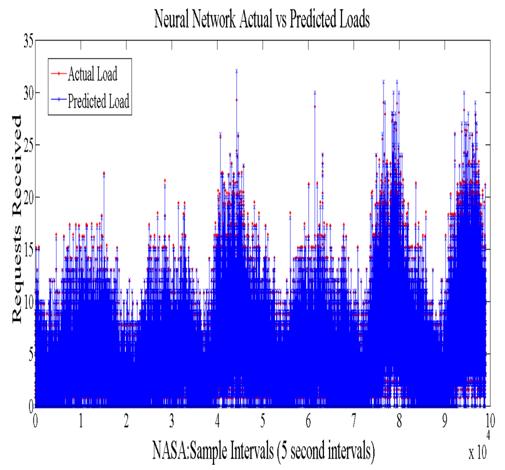

Fig.4a. NASA: Actual vs. Predicted Values for Neural Network

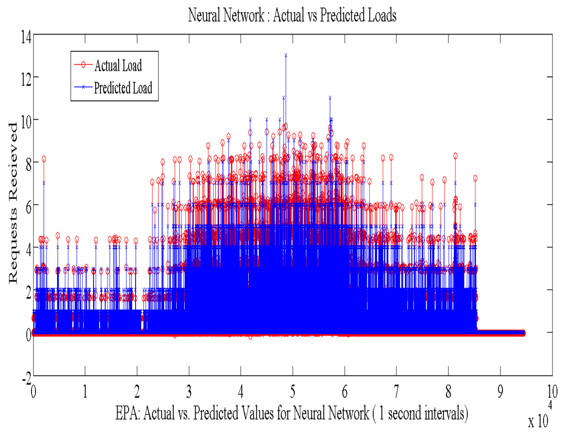

Fig.4b. EPA: Actual vs. Predicted Values for Neural Network

The Neural Network showed results consistent with prior work [6]. The Fig.4a and fig 4.b indicate a close match between the actual load value and the Predicted load Values. In fig.4a the actual load values are represented as red-circle stem plots and the predicted values as blue-x stem plots. Close examination shows that the two plots are consistently near one another.

Since there is a stochastic component to how the Neural Network trains, three separate runs were made at each Prediction Interval, and the final RMSE values were averaged.

The RMSE plots illustrated in fig 5.a and fig.5b shows approximately a linear relationship between the RMSE and the Prediction Interval. As we attempt to predict farther ahead in time, our accuracy diminishes proportional to the Prediction Interval. Linear modeling of this data can now be used to provide an estimate of the accuracy for any forecasting interval.

## B. *AR(P) Linear Prediction Simulation and Results*

The Linear Predictor uses a data window of 65 samples to calculate the optimal coefficients of the FIR filter. This allowed training to occur with a very small number of the available samples. With the filter coefficients calculated, the Weiner Filter was then used to predict the future loads at the same prediction length intervals used for the Neural Network. In practice, the training window would be repeated at discrete intervals in order to continually update the filter coefficients and therefore mitigate time-varying changes in the data.

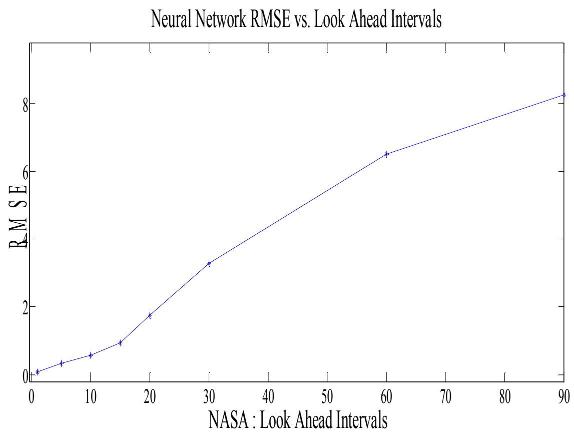

Fig.5a. NASA: RMSE per Prediction Interval for Neural Network

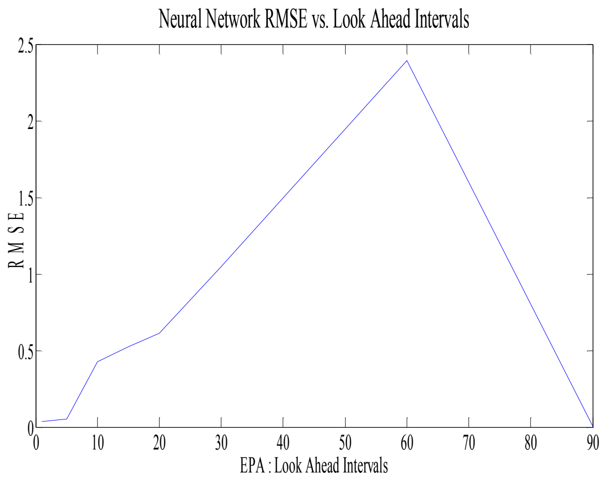

Fig.5b. EPA: RMSE per Prediction Interval for Neural Network

Next, the MSE and RMSE for each pass were calculated and the Actual vs. Predicted Load Values as well as a Squared Error dataset were recorded. The training time was relatively constant throughout each of the passes, and for 100K samples averaged 2 seconds. Fig. 6a and 6b shows the linear predictor, predicted Values vs. Actual Values of both load traces form NASA and EPA web servers.

The results of the simulation showed that the Linear Predictor modeled the reference data more closely than did the Neural Network, at all prediction intervals. This is most easily seen in the RMSE data shown in Fig.7a and fig 7b.

As is shown in fig.7a and fig 7.b, the relative values for RMSE for the Linear Predictor reach a maximum value of approximately .25 at a prediction interval of 90 seconds. The same prediction interval for the Neural Network model had an RMSE value of over 8.00. More importantly, the relationship between the RMSE and the prediction interval is again linear, which provides for a simple model for estimating RMSE at other prediction intervals.

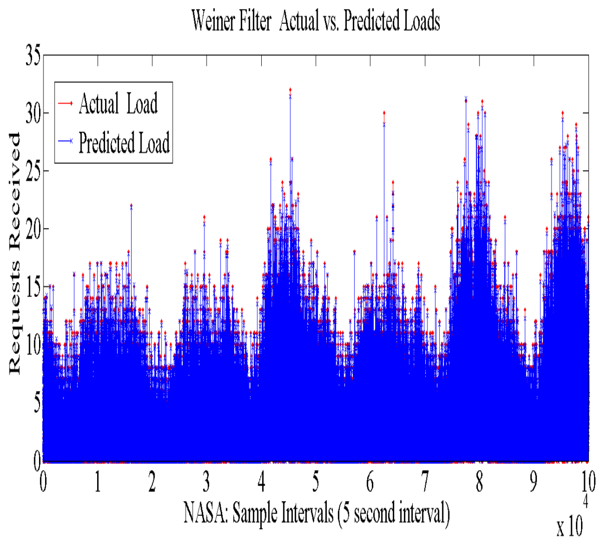

Fig. 6a. NASA : Actual vs. Predicted Values for Linear Predictor

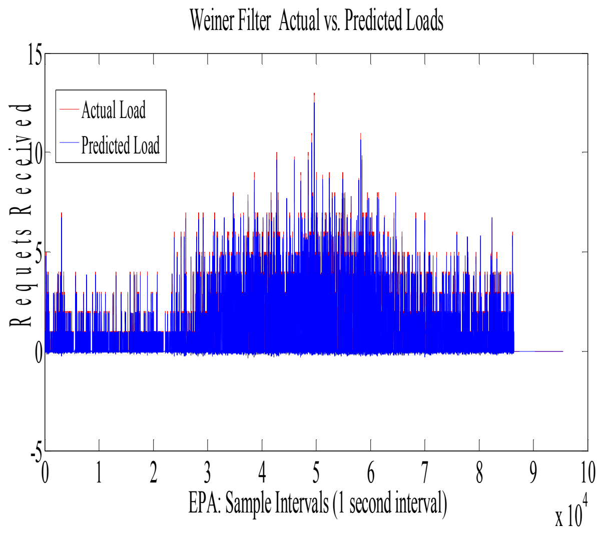

Fig. 6b. EPA : Actual vs. Predicted Values for Linear Predictor

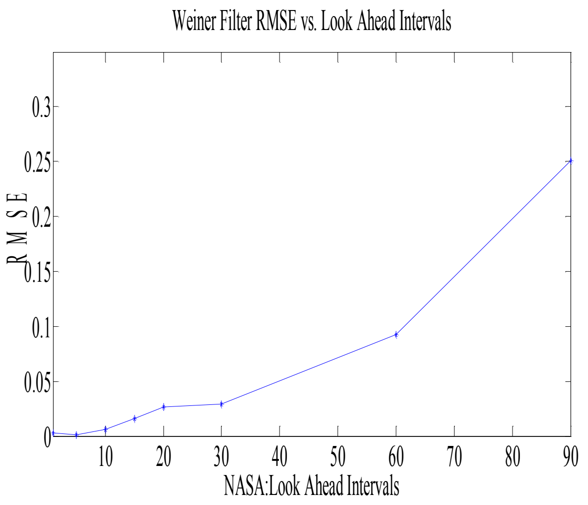

Fig.7a. NASA: RMSE per Prediction Interval for Linear Predictor

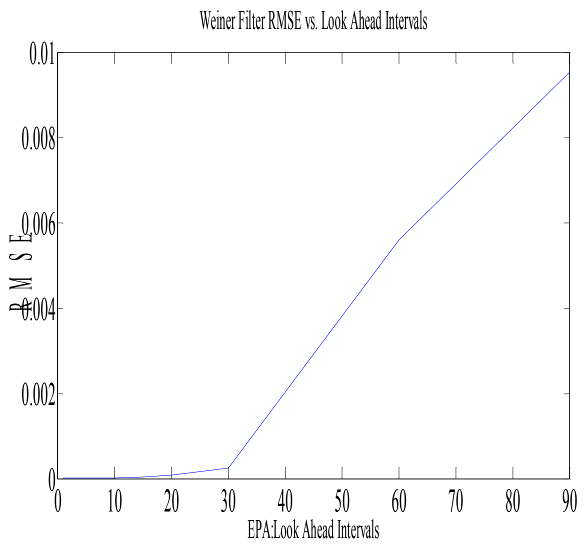

Fig.7b EPA: RMSE per Prediction Interval for Linear Predictor

## V. Conclusions

The results obtained through simulating both the Neural Network and the Linear Predictor show that both models can effectively predict future network loads, with the Linear Predictor providing the most accurate results. Furthermore, it was shown that there is an inverse linear relationship between the ability to forecast network loads and the distance into the future the load is being forecasted.

These results provide a framework that can be utilized in the determining at what confidence a cloud-controller system can manage the state-changes required for optimal operation and control.

The ultimate goal of the controller is to allocate the resources of the cloud in such a way that both an optimal power consumption level is achieved as well as maintaining all existing SLA's. By understanding the stochastic nature of the timing delays, the controller will be better able to allocate the cloud resources into their optimal system states.

## References

[1] J.G. Koomey, "Estimating Total Power Consumption byServers in the U.S. and the World".

[2] D. C. A. Jeffrey S. Chase, Prachi N. Thakar, Amin M. Vahdat and Ronald P. Doyle, "Managing Energy and Server Resources in Hosting Centers," in Eighteenth ACM symposium on Operating systems principles 2001.

[3] Kranthimanoj Nagothu, Brain Kelley, Jeff Prevost and Mo Jamshidi, "On Prediction to Dynamically Assign Heterogeneous Microprocessors to the Minimum Joint Power State to Achieve Ultra Low Power Cloud Computing", Asilomar Conference on Signals, Systems and Computers, CA , Nov-2010.

[4] B. D. Taliver Heath, Enrique V. Carrera,Wagner Meira Jr. and Ricardo Bianchini, "Energy Conservation in Heterogeneous Server Clusters," in ACM SIGPLAN symposium on Principles and practice of parallel programming, Chicago,U.S.A, 2005, pp. 186 - 195.

[5] M. Mani B. Srivastava, IEEE, Anantha P. Chandrakasan, and F. and Robert W. Brodersen, IEEE, "Predictive System Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation," IEEE Transactions on VLSI systems, vol. 4, p. 15, March 1996.

[6] Truong Vinh Truong Duy, Yukinori Sato and Yasushi Inoguchi," Performance Evaluation of a Green Scheduling Algorithm for Energy Savings in Cloud Computing", IEEE International

Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 .

[7] C.-H. H. a. A.Wu, "A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation," in International Conference on Computer Aided Design, 1997, pp. 28-32.

[8] Simon Haykins, Neural Networks : A comprehensive Foundation, Prentice Hall, 2004.

[9] Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, 2008.

[10] Danilo P. Mandic, Jonathon A. Chambers, Recurrent Neural Networks and for Prediction, John wiley&Sons, 2001Cloud Computing", IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 .

[11] Monson H. Hayes Statistical Digital Signal Processing and Modeling, John Wiley and Sons, 1996.

[12] Traces in the Internet Traffic Archive, http://ita.ee.lbl.gov/html/traces.html.