Indian Journal of Science and Technology, *Vol 8(35), DOI: 10.17485/ijst/2015/v8i35/82733, December 2015* *ISSN (Print) : 0974-6846* *ISSN (Online) : 0974-5645*

A Comparative Study on CPU Load Predictions in a Computational Grid using Artificial Neural Network Algorithms

Shaik Naseera[1*], G. K. Rajini[2], N. Amutha Prabha[2] and G. Abhishek[2]

[1]School of Computing Science and Engineering, VIT University, Vellore – 632014, Tamil Nadu, India; shaik.naseera@vit.ac.in
[2]School of Electrical Engineering, VIT University, Vellore – 632014, Tamil Nadu, India; rajini.gk@vit.ac.in, amuthaprabha@vit.ac.in, abhishek.g@vit.ac.in

Abstract

Background/Objectives: To evaluate the prediction accuracy of Neural Network algorithms for host CPU load prediction and evaluate their performance compared to actual values. **Methods/Statistical Analysis:** The speed of execution of job at the scheduled host is directly proportional to its CPU load. Therefore, target node load prediction plays an important role in job scheduling decisions. It is learnt that Neural Networks are capable of predicting the future values based on the training given on the past data. We designed a multilayer neural network and trained with learning algorithms for the input patterns collected from the load traces and predicted the future load statistics. The Mean and Standard Deviation of the predicted values are computed and analyzed against the Mean and Standard Deviation of actual values for all the ANN algorithms. **Findings:** We analyzed the prediction accuracy of Back Propagation, Quick Propagation, Back Propagation with Momentum and Resilient Propagation algorithm for the load traces collected from variety of computers connected in a network. Existing reports shows that Back Propagation algorithm exhibits better prediction accuracy compared to statistical approaches like linear regression and polynomial regression. In this paper, we have shown that Resilient Propagation algorithm has better prediction accuracy compared to other ANN algorithms. **Application/Improvements:** Job scheduling and resource selection algorithms can employ neural network algorithms to predict the load for the sharable resources connected in the network for more accurate and faster scheduling/resource selection decision.

Keywords: CPU Load Prediction, Load Traces, Neural Network Algorithms, Training, Resource Selection

1. Introduction

The prediction of CPU load at the remote host is essential to address the following issues. First, in a computational grid[1], the jobs migrate from one host to another host to make use of the required resources for its execution. The reason behind the job migration in a computational grid is due to the non-availability of the resource at the source node. Second, as a means of load balancing policy in the grid, the jobs migrate from one host to another host for effective utilization of the under-utilized resources, there by improving the overall throughput of the grid system. Third, the network servers that provide services for the client requests need to enhance their capability during their peak loads to cope up with the increasing demand. The capability of such servers are improved by means of hardware redundancy. The prediction of host CPU peak load timings and duration is helpful to overcome the server crashes or overload bottlenecks. Because, the execution time of a job is found to be linear to the CPU load of the chosen host[2,3]. Fourth, the network servers need to undergo service and maintenance periodically and hence need to shutdown for certain duration of time. In that case, the load prediction helps to predict the less load timing intervals of the server and hence reduce the dis-satisfaction among the clients waiting for the service.

** Author for correspondence*

In this paper, we evaluate the performance of different artificial Neural Network (ANN) algorithms for accurate host CPU load prediction. We compare the performance of different ANN algorithms on different network architecture for the variety of load traces given by Dinda[4,5].

2. Related Work

Researchers have proposed several prediction strategies for the CPU load prediction in the grid environment[6,7]. Network Weather Service (NWS), is the most popular among them. In a given time interval, NWS dynamically forecasts and periodically monitors the performance of the network. NWS uses different predictive methods to measure the performance for the next time interval. Sliding window average, Running average, Adaptive window average, Last measurement, Adaptive window median, Median filter, Stochastic gradient, α-trimmed mean and Auto-regression are the predictive methods used in NWS.

Dinda and Hallaoron[8] evaluated different linear time series models[9] like MA, AR, ARMA, ARFIMA and ARIMA, for predicting loads for the future. The authors have proved that prediction accuracy and overhead model like AR is sufficient for CPU load prediction.

Homeostatic and tendency-based one-step-ahead prediction strategies are proposed by authors[10]. The mean of a time series remains steady in homeostatic methods where in the tendency of the time series retains in tendency-based strategies. i.e., if current value increases, the next value will also increase, or vice versa. It is observed that these techniques perform better than the methods used in NWS.

Generally, every CPU-bound task execution time is directly related to the CPU load of host node. Sometimes the measured load and the execution time of the CPU bound task is directly proportional to each other[8]. Therefore, if we predict the load of the host node, we could estimate the execution time of the task on that machine. Hence, load prediction is useful for guiding the scheduling strategies to achieve high application performance and for efficient resource usage[2,3,11,12].

Dinda[5] provided a comprehensive analysis, including distributions, summary statistics and time series analysis on the host load statistical properties. Experimental results given in literature[13-16] shows that tendency-based prediction has better performance over NWS. Work done by authors[17,18] shows that ANN algorithms are well suitable techniques for pattern recognition and classification.

3. Neural Network based Prediction Strategies

ANN mimics the human brain learning methodology. ANN is a non-algorithmic based approach for processing information. ANN is like an efficient data-modeling tool for capturing and representing complex input-output relationships. The motivation behind the development of ANN is to develop an artificial system that performs intelligent tasks similar to the tasks performed by the human brain.

3.1 Advantages of ANNs

An ability to learn based on training, ability to create own representation, Parallel computation, Fault tolerant capability are the feature which make ANN's more efficient.

3.2 Learning Phase in ANNs

Learning is a process by which the free parameters of a ANN are adapted through a process of stimulation by the environment in which the network is embedded. The learning methods used for ANNs can be classified into two major categories: *Supervised learning* and *Unsupervised Learning.* In this paper, we designed a multi-layer ANN 10×15×10×1 as shown in Figure 1 for the purpose and we have chosen supervised learning for the training.

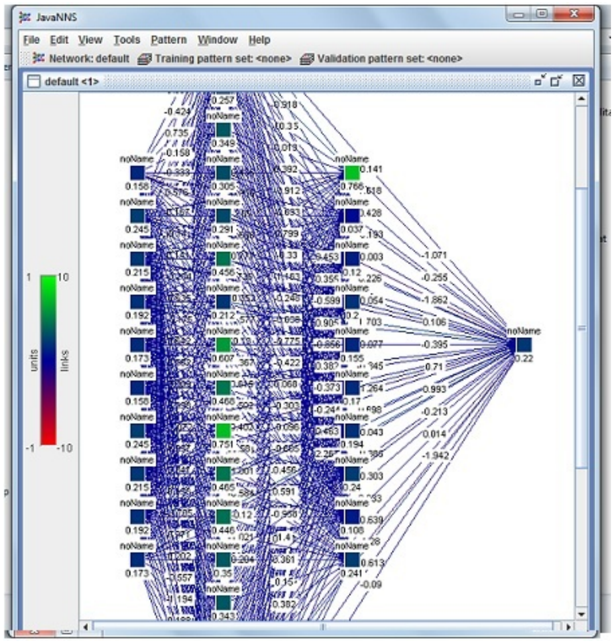

JavaNNS
File Edit View Tools Pattern Window Help
Network: default
Training pattern set: <none>
Validation pattern set: <none>
default <1>
units
links
noName

Figure 1. A Multi-Layer ANN design using JavaNNs software.

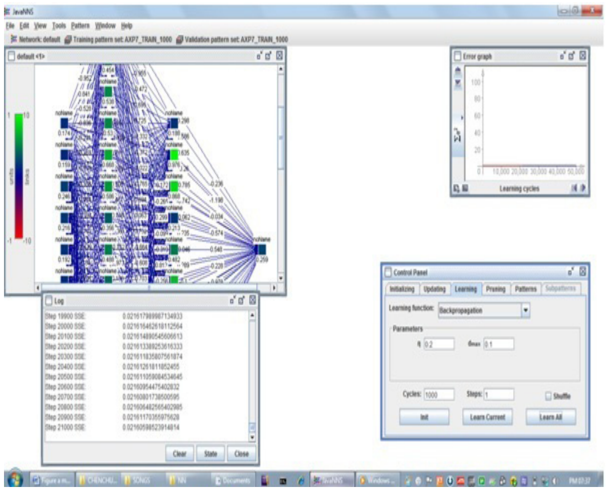
3.2.1 Training Phase

We have done the training for the ANN shown in Figure 1 separately by using the learning algorithms listed below and the load predictions are recorded and tabulated.

- Back-Propagation (BP) Algorithm
- Back-Propagation with Momentum (BPM)
- Batch Back-Propagation (BBP)
- Quick-Propagation (QP)
- Resilient-Propagation (RP)

3.3 Algorithm for Supervised Learning

Input:
A data set consisting of the training tuples and their associated target values;
A learning rate parameter;
A multilayer feed-forward network.
Output:
A trained ANN



```
Step 19900 SSE:   0.0216179898713493
Step 20000 SSE:   0.0216164626181125
Step 20100 SSE:   0.0216148905456066
Step 20200 SSE:   0.0216133882536163
Step 20300 SSE:   0.0216118358075618
Step 20400 SSE:   0.021612618118524
Step 20500 SSE:   0.0216110590845346
Step 20600 SSE:   0.021609544754028
Step 20700 SSE:   0.021608017385005
Step 20800 SSE:   0.0216064825654029
Step 20900 SSE:   0.021611703655975
Step 21000 SSE:   0.021605985239148
```

Learning function: Backpropagation
Parameters: η 0.2   dmax 0.1
Cycles: 1000   Steps: 1   Shuffle

Figure 2. A Trained ANN using Back Propagation Training Algorithm.

Method:

- Initialize all weights and biases in the network;
- The input vector containing the training patterns is given to the input layer of the ANN. The inputs propagate through the ANN till to the output layer. This is called the forward pass and the output layer produces the actual or predicted value.
- The desired outputs are also given in the training patterns. The actual network outputs are subtracted from the desired outputs and an error is calculated.
- This error is the basis for the backward pass. Here, the errors are passed back through the ANN by computing the contribution of each neuron present in the hidden layer and the corresponding adjustment is carried out in the backward pass. The connection weights are thus adjusted accordingly and hence the ANN learn from an experience.

Two major learning parameters are used to control the training process in any supervised learning algorithm: learning rate and output error. The learn rate is used to specify whether the ANN is going to make major/minor adjustments after each learning cycle. Output error is used to check whether the training is completed or not.

4. Experimental Environment and Performance Analysis

A series of experiments are conducted for the prediction algorithms listed in Sec 3.3 and evaluated its effectiveness for the load traces. The loads traces we used here are the load traces collected by Dinda[7] on a group of machines. Two different groups of machines are considered: Alpha cluster and Computer server with a desktop workstations.

The properties of these machines are given in of this group, the machine *axp0* is an interactive machine, while *axp7* is a batch machine. The machines in the second group are a computer server (*sahara*) and a desktop workstation (*themis*).

Axp0. psc. edu is a heavily loaded machine with mean CPU load of 1 and standard deviation 0,54. It is computed for the data collected for 75 days (1,296,000).

Axp7. psc. edu is a lightly loaded machine with mean CPU load of 0.12 and standard deviation 0.14. It is computed for the data collected for 65 days (1,123,000).

Sahara. cmcl. cs. cmu. edu is a moderately loaded machine with mean CPU load of 0.33 and standard deviation 0.14. It is computed for the data collected for 20 days (3,45,600).

Themis. nectar. cs. cmu. edu is a moderately loaded desktop machine with mean 0.49 and high standard deviation of 0.5. The total number of data in this time series is for 20 days (345,600).

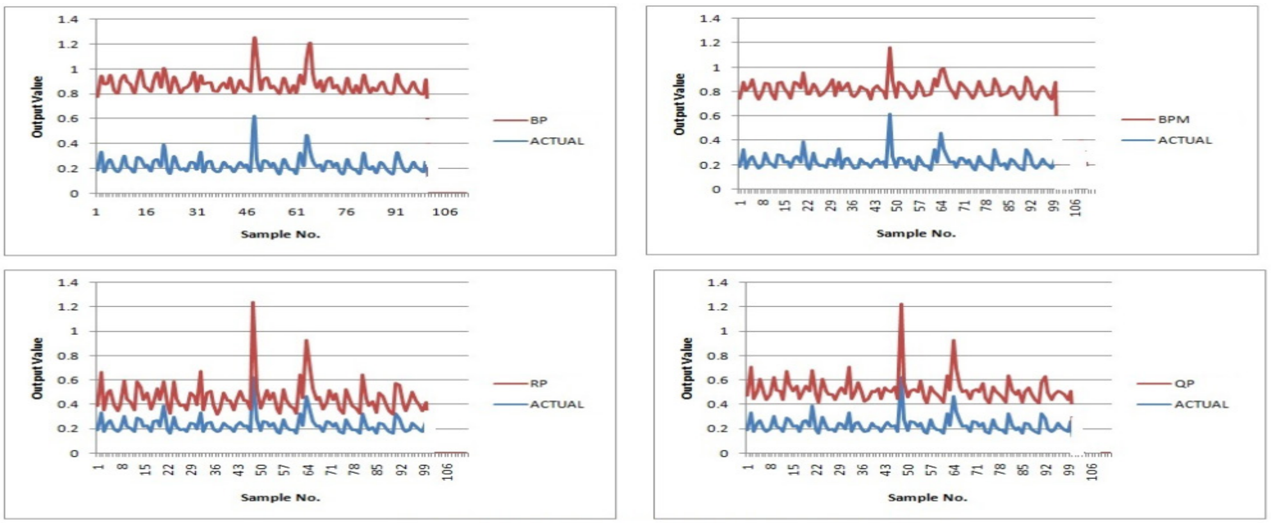


Figure 3. Graph of Actual Vs. Predicted values from ANN Algorithms.

Table 1. Actual Vs Predicted Values from ANN Algorithms

ACTUAL	Actual Value Vs Predicted Value				
	BP	BPM	BBP	QP	RP
0.186967	0.59289	0.55593	0.37537	0.2909	0.20204
0.331748	0.61001	0.54366	0.42465	0.37491	0.33722
0.173132	0.70846	0.63822	0.3699	0.27251	0.18391
0.247251	0.63879	0.59499	0.35325	0.26068	0.24911
0.268005	0.68184	0.62854	0.41251	0.34293	0.25243
0.197344	0.63644	0.57589	0.36549	0.29706	0.20261
0.174614	0.6301	0.56094	0.38839	0.26588	0.17549
0.200803	0.70371	0.60239	0.35091	0.27907	0.20616
0.300124	0.65366	0.57258	0.41582	0.32018	0.29423
0.221557	0.68093	0.64481	0.32432	0.29748	0.22319
0.204262	0.66717	0.5859	0.36408	0.30668	0.21121
0.175108	0.6328	0.56179	0.38842	0.26704	0.17706
0.285794	0.61168	0.58643	0.49241	0.38682	0.30183
0.277888	0.71214	0.60169	0.34076	0.28734	0.26206
0.219086	0.63957	0.61131	0.34689	0.28899	0.22345
0.22551	0.62031	0.56108	0.40223	0.32979	0.27292
0.177085	0.64177	0.56462	0.38825	0.27025	0.18184
0.258122	0.64711	0.61898	0.37205	0.25249	0.1629
0.271464	0.69933	0.6011	0.35006	0.27866	0.26059
0.218592	0.63657	0.61041	0.3474	0.28828	0.22246
0.39055	0.61592	0.56929	0.36644	0.29012	0.19425
0.191414	0.70714	0.58616	0.38484	0.28994	0.20283
0.160778	0.65314	0.62082	0.37228	0.25489	0.16592

Dinda collected the CPU load samples at some point periodically and exponentially averages some previous samples to produce a load average. 1Hz sampling rate is used to satisfy the Nyquist criterion and thus captures all of the dynamics of the system load.

We have trained the ANN shown in Figure 1 with load traces given by Dinda[15] for 20,000 cycles to obtain an error rate of 0.02 using Java NNs software. Figure 2. shows the resultant trained ANN, learning rate parameter value, error graph and error log generated during training for back propagation algorithm. The similar procedure was adapted for the remaining training algorithms. Likewise, we have done the training of ANN for all the algorithms listed in Sec. 3.3. and the predictions are recorded and tabulated. The Table 1. shows the sample predicted values collected from the trained ANN using different prediction algorithms. The graph showing the differences in predicted and actual values for all the ANN algorithms are given in Figure 3. The Mean and Standard Deviation of the prediction error for the ANN algorithms are shown in Figure 4. It is observed that BP has more prediction error and RP has least prediction error among the other ANN algorithms chosen in this paper.

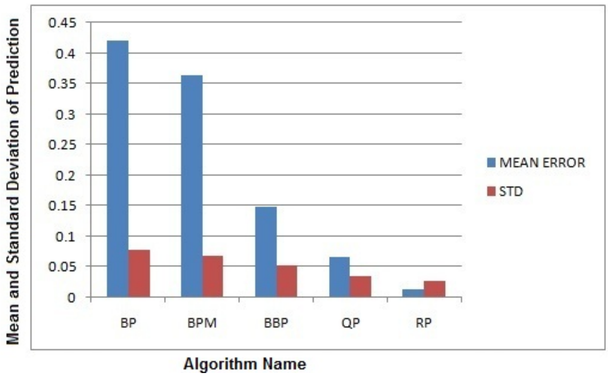


Figure 4. Mean and Standard Deviation of prediction error for different ANN algorithms.

5. Conclusions and Future Scope

In this paper, we have compared the prediction accuracy of different neural network training algorithms. The experimental results show that Resilient Propagation exhibits better performance and prediction accuracy among the other ANN training algorithms.

We plan to extend our work for the prediction of network bandwidth and latency. By prediction this information, one can guide the scheduler for efficient data placement on a wide area network.

6. References

1. Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, San Fransisco, CA, 1998.
2. Yang L, Schopf JM, Foster I. Conservative scheduling: Using predicted variance to improve decisions in dynamic environment. Supercomputing'03. 2003; 31.
3. Jang S, Wu X, Taylor V. Using performance prediction to allocate grid resources. Technical report, GriPhyN. 2004-25; 1–11.
4. Load Trace Archive. Available from: http://www.cs.cmu.edu/~pdinda/LoadTraces/
5. Dinda PA. The statistical properties of host load. Technical report, CMU. 1998; 1–23.
6. Akioka S, Muraoka Y. Extended forecast of CPU and network load on computational grid. IEEE Int'l Symp on Cluster Computing and the Grid. 2004 Apr 19-22; 765–72.
7. Liang J, Nahrstedt K, Zhou Y. Adaptive Multi- Resource Prediction in Distributed Resource Sharing Environment. IEEE Int'l Symp on Cluster Computing and the Grid. 2004 Apr 19-22; 293–300.
8. Dinda PA, O'Hallaron DR. Host load prediction using linear models. Cluster Computing. 2000 Dec; 3(4):265–80.
9. Box G, Jenkins G, Reinsel G. Time Series Analysis Forecasting and Control, Prentice Hall, 1994.
10. Yang L, Foster I, Schopf JM. Homeostatic and tendency-based CPU load predictions. Int'l Parallel and Distributed Processing Symp. (IPDPS'03). 2003 Apr 22-26; 42–50.
11. Dinda PA. A prediction-based real-time scheduling advisor. Proceedings of 16th Int'l Parallel and Distributed Processing Symp (IPDPS 2002). 2002. p. 35–42.
12. Lu D, Sheng H, Dinda P. Size-based scheduling policies with inaccurate scheduling information. 12th IEEE Int'l Symp on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04). 2004 Oct 4-8; 31–8.
13. Swany M, Wolski R. Multivariate resource performance forecasting in the network weather service. Supercomputing'02. 2002; 1–10.
14. Wolski R, Spring N, Hayes J. Predicting the CPU Availability of Time-shared Unix Systems on the Computational Grid. Proceedings of 8th IEEE Symp on High Performance Distributed Computing. 1999; 105–12.
15. Wolski R. Dynamically forecasting network performance using the network weather service. Journal of Cluster Computing. 1998 May; 1(1):119–32.
16. Wolski R. Experiences with predicting resource performance on-line in computational grid settings. ACM SIGMETRICS Performance Evaluation Review. 2003 Mar; 30(4):41–9.
17. Zebardast B, Maleki I, Maroufi A. A noval multilayer perceptron artificial neural network based recognition for kurdish manuscript. Indian Journal of Science and Technology. 2014 Mar; 7(3).
18. Gharehchopogh FS, Khaze SR, Maleki I. A new approach in bloggers classification with hybrid of K-nearest neighbor and artificial neural network algorithms. Indian Journal of Science and Technology. 2015 Feb; 8(3).