Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp407-410)

Performance Prediction of a Parallel Monte Carlo Application: A Neural Network Approach

SIRMA YAVUZ
Computer Engineering Department
Yıldız Technical University
34349 Beşiktaş, İstanbul
TURKEY
http://www.ce.yildiz.edu.tr/personal.php

Abstract: - A feedforward neural network model is presented in this study to predict the execution time of a parallel Monte-Carlo implementation. The enormous performance range offered by today's systems caused the performance evaluation tools to become more complicated to be able to consider the relative values and interrelated parameters. Artificial Neural Networks provide an excellent alternative to conventional techniques with their ability to capture many kinds of relationships and have been used successfully in various prediction tasks. However, their use in performance prediction area is a novel approach. The Neural Network model proposed here is aimed to be simple, general and reliable. This work also demonstrates the potential of artificial neural networks in identifying the contribution of interrelated system and application parameters to performance. Prediction of computational and communication execution times of the application is examined in this paper.

Key-Words: Feedforward, Backpropagating, Neural Network, Performance Prediction, Parallel Systems, Monte Carlo

1 Introduction

Conventional performance evaluation techniques include benchmarking, analytical modeling and simulation [1]. Each of these methods has their advantages as well as limitations. Analytical models are very powerful but their applicability is not universal. In order to be traceable, they do simplifying assumptions about the architecture and application characteristics that may not reflect the accurate representation of reality. While simulation tools allow designers to understand the system beyond analytical models can provide, they are expensive to run and do not replace real measurements. Benchmarking is very popular, but often they do not use real world applications and open to misuse. The common methodologies used in benchmarking and the pitfalls encountered are explained in [2].

On the other hand, ANNs provide flexible exciting alternatives to the conventional methods [3]. Usually neural networks complement the conventional techniques and in literature they have been successfully used for numerous prediction tasks, varying from financial time series prediction to river flow modeling [4, 5].

2 Specifications of the Test Platforms

Numbers of SunSparc workstation clusters were used to collect the training,validation and test data, measuring the execution time of the parallel Monte-Carlo implementation on different workstation clusters containing between 2 to 256 processors each time.

The specifications of these workstations are given in Table 1.

Table 1. Specifications of the Tested Workstations

	SparcStation 5	**Ultra 1**	**Ultra 10**
Architecture	**UltraSPARC (MicroSPARC-II compatible)** SPARC V-8	**UltraSPARC-I** SPARC V-9	**UltraSPARC-II i** SPARC V-9
CPU Clock Rate	170 MHz	143 MHz	300/333 MHz
Main Memory	64 Mb	64 Mb	128 Mb
L1 Cache	16Kb Data 16Kb Instruction	16Kb Data 16Kb Instruction	16Kb Data 16Kb Instruction
L2 Cache	512Kb	512Kb	2Mb
Addressing/Data	32 bit	64 bit	64 bit
Functional Units	Optimized Integrated FPU. 1 Integer Unit	3 FPU Execution Units. 4 Integer Execution Units with 2 ALUs.	3 Floating Point Execution Units. 4 Integer Execution Units with 2 ALUs.
Pipeline Stages	5 Stage Pipeline can issue up to four instructions per cycle	9 Stage Pipeline can issue up to four instructions per cycle	9 Stage Pipeline can issue up to four instructions per cycle

To utilize the Network of Workstations as a parallel resource, MPICH, a complete implementation of the Message Passing Interface MPI, has been used [6, 7].

3 Monte Carlo Application

Monte Carlo methods are effective for pricing some financial derivatives, especially when the price cannot be calculated analytically because it depends on the historical movement of underlying variables [8].

The parallel application studied here, calculates European option prices using Monte Carlo simulation techniques. The set of trajectories were divided equally among the processors, and the results reported back to a master. The parallelization has the following form:

- Each processor calculates the trajectories of asset prices,
- the master sums the individual results,
- each slave obtains the sum from the master and calculate local error,
- finally the master sums the individual errors.

The accuracy of the resulting price is primarily a function of the number of Monte-Carlo trials performed.

4 Architecture of the Neural Network

The model presented in this study is a multilayer feedforward network that is used with the backpropagation algorithm. Two separate models, with same architecture and different inputs, have been used to predict the computational and communication performance of the Monte Carlo application.

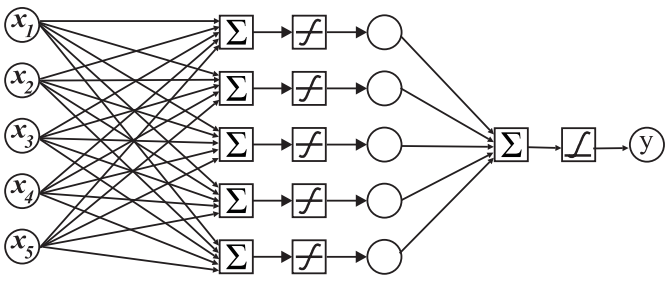

Fig. 1. Architecture of the computational and ommunication performance prediction model

The selected training algorithm is the Levenberg-Marquardt, which is one of the most successful non-linear curve fitting methods [9]. Its utility is the way in which the fit method moves smoothly between the two extremes, the steepest descent and the Hessian, for finding the next step size. The algorithm uses the method of steepest descent to determine the step size when the results are far from the minimum.

The implementation has been successfully tested on a large number of nonlinear problems. It has proved to be more robust than the Gauss-Newton method and iteratively more efficient than an unconstrained method.

Fig.1 shows the architecture of the computational and communication performance prediction model. The activation function for the hidden units is the tansig function defined as $f(x) = \frac{2}{1+e^{-2x}} - 1$, while the output activation function is the logsig function defined as $f(x) = \frac{1}{1+e^{x}}$

The input parameters of the network are basic architectural and application parameters which can be obtained from the data sheets and from the source code of the application. The inputs for computational performance prediction:

- *Number of Processors*: The number of processors on the network running the target application.
- *CPU Speed*: The CPU speed of the microprocessors running the application.
- *Problem Size*: For the selected Monte-Carlo application, problem size includes the *Number of Trials* and the *Number of Varietes*.
- *Number of Integer Operations*: This is the number of integer arithmetic operations in the application, determined from the source code of the application.
- *Number of Floating Point Operations*: This is the number of integer arithmetic operations in the application, determined from the source code of the application.

The inputs for communication performance prediction:

- *Number of Processors*
- *CPU Speed*
- *Problem Size*
- *Number of MPI Communication Calls:* The number of MPI point-to-point communication calls in the, determined from the source code of the application.
- *Network Bandwidth*: The maximum rate at which the interconnection network can propagate information, once the message enters the network.

5 Data Analysis and Processing

The quality of the predicted sequence is important to train the network properly and for the success of the

prediction task. Very often in datasets, samples that do not comply with the general behavior or model of the data exist. Such data samples, which are extremely different from or inconsistent with the remaining set of data, are called outliers [10]. The dataset used in this study have been checked and does not contain any extreme outliers. The training and test data has been also verified for randomness, using the 'Run' test [11].

There are many techniques and considerations relevant to data pre-processing. Pre-processing can vary from simple filtering (as in time-series data), to complex processes for extracting features from image data. In this study min-max normalization is applied to squash the values to the intervals [0, 1].

Data post-processing covers any process that is applied to the output of the network. As with pre-processing, it is entirely dependent on the application and sometimes it is just the reverse process of data pre-processing, as in this study.

The division of the data into the training,validation and test sets is an important issue as well. It is critical to represent the population and the underlying mechanism in both the training and the test sets. For nonlinear forecasting models, it is suggested to use at least the %20 of the data for testing [12].

6 Summary and Results

To train a network and measure how well it performs, an error function (or cost function) must be defined. The weights of the network are updated in the direction that makes the error function minimum. The mean square error (mse), one of the most common error functions, is used in this study and given in Equation (1). The error is calculated as the difference between the target output t and the network output a. Weights are adjusted to minimize the average of the sum of these errors.

$$mse = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q} (t(k) - a(k))^2 \qquad (1)$$

Also average percentage error and the standard deviation are calculated for the data set using the Equations (2) and (3), respectively. Q is the number of elements in the data sets. The results are given in Table 2.

$$Avarage Error(\%) = \frac{1}{Q}\sum_{i=1}^{Q} \frac{|Measurement_i - Prediction_i|}{Meeasurement_i} x100\% \qquad (2)$$

$$Stdev = \sqrt{\frac{\sum_{i=1}^{Q} (Error(\%)_i - AverageError(\%))^2}{Q-1}} \qquad (3)$$

Table 2. The Average Error and the Standard deviations for the selected datasets

	Dataset1 contains MC communication test data	Dataset2 contains MC computation test data
Number of training data	220	140
Number of test data	53	60
Number of Validation data	27	20
Network performance	1.418E-07	2.15E-04
Training performance	1.354E-07	1.00E-07
Average Error (%)	0.19110	0.1885
Standard Deviation	0.13	0.089

The prediction results for the computational prediction model and the squared errors during training are given in Fig. 2 and Fig. 3, respectively.

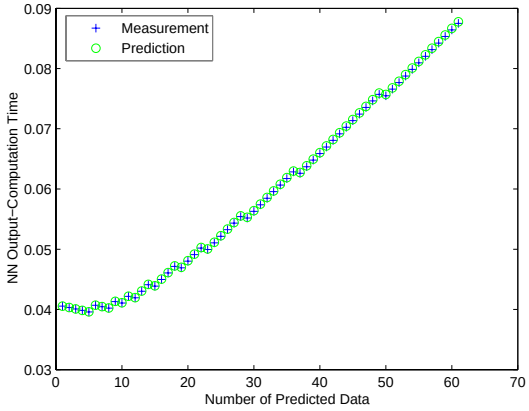

Fig.2. Network performance for Dataset 1

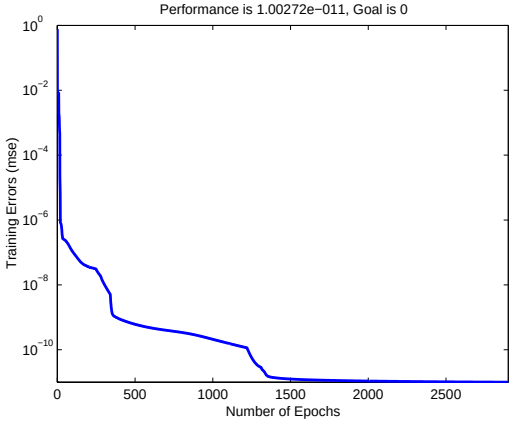

Fig.3. Squared errors during training for the Dataset 1

The prediction results for the computational prediction model and the squared errors during training are given in Fig. 4 and Fig. 5, respectively.

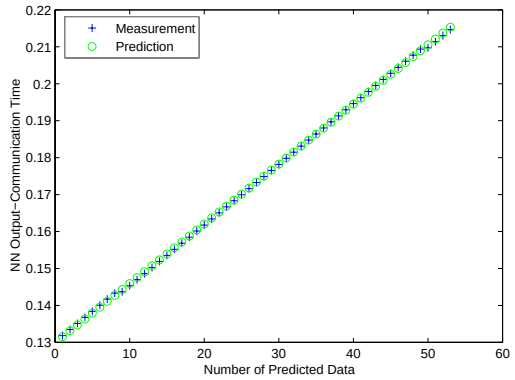

Fig.4. Network performance for Dataset 2

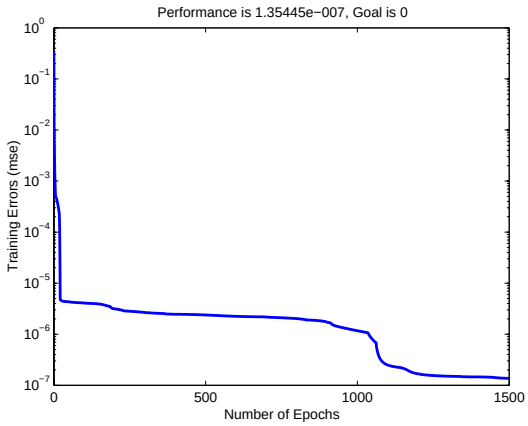

Fig.5. Squared errors during training for the Dataset 2

7 Conclusions

Although, the artificial neural networks have been used for various prediction tasks, their use in performance prediction area is novel. A two-layered feedforward neural network model that takes application and hardware parameters as its inputs, to predict the execution time of the selected application on a target system, is presented in this paper.

In particular, our predictions are within approximately 5% of measured execution times. Almost a perfect fit between measurements and predictions have been achieved in most cases.

However, determination of the presented neural network inputs requires a level of effort, in terms of identifying the number of operations and number of MPI communication calls, which are application dependent. These parameters have been expressed as a function of problem size and number of processors, after an in-depth analysis of target application's source codes.

8 Acknowledgements

The measurements are taken on SunSparc workstations, existed at the Computer Science Department, University of Warwick, UK.

References:

[1] Jain R.,*The Art of Computer Systems Performance Analysis-Techniques for Experimental Design, Measurement, Simulation and Modelling*, John Wiley & Sons Ltd, 1991

[2] Dongarra J.J., Martin J., and Worlton J, *Computer Benchmarking Paths And Pitfalls*, IEEE Computer 24, pp 38-43, 1987

[3] Jain Anil K., Mao J., Mohiuddin K. M., *Artificial Neural Networks: A Tutorial*, IEEE Computer Vol 3, No 29,pp 31-44, 1996

[4] Neji Z., and Beji F.M., *Neural Network and Time Series Identification and Prediction*, IEEE INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), 2000

[5] Atiya A. F., El-Shoura, S. M., Shaheen, S. I. and El-Sherif. M. S., *A Comparison Between Neural-Network Forecasting Techniques-Case Study: River Flow Forecasting,* IEEE-NN, Vol 10, 1999

[6] Pacheco Peter S., *Parallel Programming with MPI*, Morgan Kaufmann, 1997

[7] Gropp W.,Lusk E., and Skjellum A., *Using MPI: Portable Parallel Programming with the Message Passing Interface*, The MIT Press; 2nd edition, 1999

[8] Jackel P., *Monte Carlo Methods in Finance*, John Wiley & Sons Ltd., 2002

[9] Reed R. D. and Marks R. J., *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, MIT Press, 1999

[10] Han J. and Kamber M., *Data mining concepts and techniques,* Academic Press, San Francisco, 2001

[11] Knuth E. D., *The art of computer programming*, 2 nd edn, Addison-Wesley, 1981

[12] Granger C.J.W., *Strategies for Modelling Non-linear time-series relationships*, The Economic Record, pp 233-238