Developing Multilayer Neural Network Applications on a Distributed Memory Parallel Machine.

Eugenio Suárez Caner
COPPE/EE/UFRJ, CP 68504, Rio de Janeiro 21945-970, Brazil
CEADEN, CP 6122 ,La Habana, Cuba
e-mail:eugenio@lps.ufrj.br

Jose Manoel de Seixas
COPPE/EE/UFRJ, CP 68504, Rio de Janeiro 21945-970, Brazil
e-mail: seixas@lps.ufrj.br

Abstract

This paper concern the development of multilayer artificial neural network applications in a transputer (T9000) based parallel machine with distributed memory. For defining the partition of processing tasks within the machine, the natural parallelism exhibited by neural network is explored. Both training and production phases can be implemented and the designer defines the set of parameters required by the backpropagation training method through an user interface. Neural preprocessing methods based on topological mapping and principal component analysis are also available to be integrated into a neural network based hybrid system design. As a case study, the principal component analysis for a particle discriminator in experimental physics is developed.

1 Introduction

Artificial neural networks (ANNs) have been attracting a considerable attention as problem solvers of a wide range of real-world problems. One important feature that has been explored quite successfully on these networks is their inherent parallelism, which allows to perform nonlinear adaptive processing at high speeds. Thus, many ANN based real-time systems are found in practice and the number of such applications tends to increase, as the role of ANNs in such systems is being updated [1].

In this paper we focus on feedforward multilayer artificial neural networks that are trained by the backpropagation algorithm [2]. This supervised learning algorithm and such multilayer topology deserve such attention, as the majority of the neural applications involve the combination of the two. On the other hand, as one crucial factor of such applications is the long time required by the learning process, specially when the ANN models become large, the work to be presented develops neural processing applications in a parallel and highly connected machine.

A 16-node transputer based parallel machine (TN-310 system) was selected as the development platform. This multiple instruction multiple data (MIMD) parallel computer also employs a fast digital signal processor (DSP) running as a coprocessor to the transputers, for optimizing signal processing applications.

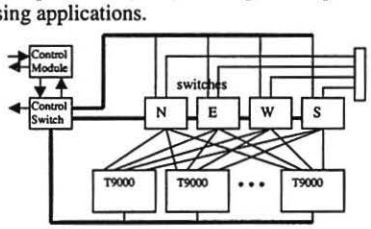

Figure 2.1 The basic architecture of the TN-310.

2 TN-310 System

The basic architecture of the TN-310 [3] system is shown in Figure 2.1. It consists of 16 processing nodes that combine T9000 transputers, for managing efficiently the communication between nodes, and ADSP-21020 digital signal processors, for running signal processing applications at high speed. In terms of memory, each node comprises 256 kbytes SRAM used as shared memory, to transfer data to and from the DSP and 8 Mbytes of T9000

private DRAM. The transputer can run a number of concurrent processes and the machine being used supports the complete communication among processing nodes by means of a network of C104 chips. DSP can be programmed from the T9000 through C runtime library calls.

3 Parallel Training of Feed-forward Networks

The back-propagation algorithm and the structure of feed-forward networks, respectively, include a number of different schemes of parallelism. In [6] the principal dimensions of ANN parallelism are listed. In case of the simulation on MIMD-computers we only take sample parallelism and unit parallelism into account.[7]

Sample Parallelism: The gradient vector estimation is a result of adding its partial components, which correspond to the patterns visited in an epoch, from a training procedure in batch mode. In sample parallelism (see Figure 3.1) each slave has a complete copy of the neural net and calculates part of gradient estimate, while the master accumulates the partial estimates and uses the complete gradient estimate to update the weights on the networks running in the slaves. Sample parallelism is more adequate for training procedures based on large batch sizes.

Unit Parallelism: This fine-grain approach partitions the neural net so that each processing node gets about the same quantity of synaptic weights. That is, each processor computes part of the intermediate values during forward and backward phases of the learning procedure for a training set. Figure 3.2 sketches this approach. Unit parallelism is rather communication intensive compared to sample parallelism and is more effective for large networks.

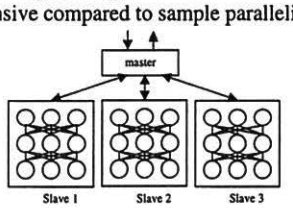

Figure 3.1 Sample Parallelism.

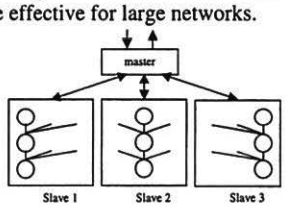

Figure 3.2 Unit Parallelism.

4 Communication Task and Mapping on TN-310.

Transputer machines can be configured in a variety of topologies. The TN-310 is an *ether* (ideal) network, i.e. all processes directly communicate to each other ("any to any"), and it is also a pseudo-dynamic network, in the sense that the topology is modified just before the execution time by using programmable link crossover switches. These features are explored in this work to configure interprocessor communication according to the parallelization methods described in the last section.

As examples, Figure 4.1 shows the mapping for unit (a) and sample (b) parallelism respectively. For each mapping, the main features of the communication network and the specific architecture of the TN-310 system are taken into account. By assigning to neighbor nodes processing tasks that require intensive data transmission among them, the communication overheads can be minimized. Therefore, in Fig. 4.1, the first parallel processing nodes that are to be selected for an application are the ones, which communicate through a single C104 switch. The arrow in this figure indicates the selection criterion.

To illustrate this node allocation criterion, an application of unit parallelism which requires only 4 parallel processing nodes would be allocated to nodes [1]-[4] (in the same processing board), whereas the distributor (master) node is placed to run on node 0. As this parallelization method demands intensive communication among the processors that work in parallel, parallel processing nodes that require a single C104 to communicate are preferable.

On the other hand, sample parallelism is characterized by intensive communication between the master node and slaves, so that the master nodes is allocated to node [1] (node [0] requires at least two C104 in a routing) and nodes [2]-[5] form the best set for performing parallel processing for such 4 node application.

Figure 4.1 The mapping for unit (a) and sample (b) parallelism.

5 Case Study: Principle Component Analysis for a Particle Discriminator

The principal component analysis (PCA) is known as a technique that can perform dimensionality reduction with preservation of the information spread around the full data input space [2,6]. This neural principal component analysis was applied successfully to a particle discrimination problem in high-energy physics [11]. For a collider experiment that will be operational by the year 2005, the Large Hadron Collider, an online validation system is being designed for triggering on very rare events which are of interest in the experiment but are masked by the huge background noise generated by the colliding machine.

Such neural principal component analysis was implemented in the TN-310 system. Both sequential and unit parallelism methods were used. Knowing the communication features of the TN-310 system, the implementation methodology searched for optimizing the processor to node allocation scheme (Fig. 4.1), the communication path and the level of parallelism.

As a single neural network approach to the particle discrimination problem achieved good classification efficiency with ten nodes in the hidden layer, Figure 5.1 shows a simulation result of a *121-i-121* network implementation, indicating how the time required for a training cycle (one forward signal propagation followed by a backward propagation of the error signal) varies as the number of parallel nodes (the number of partition layers in the unit parallelism approach) and the network complexity (the number of *i* hidden nodes) are increased. These curves are obtained from a piecewise linear approximation of the function, which describes the communication time for an increasing data size to be transmitted through the switching network. Figure 5.2 shows how the speedup factor varies with the number of parallel processing nodes used for this application. It can be clearly seen in this figure that communication overheads decrease the speedup factor when an excessive number of partition layers is used.

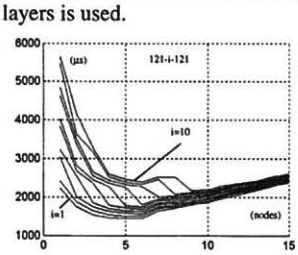

Figure 5.1 Optimal partition of the processing task in unit parallelism.

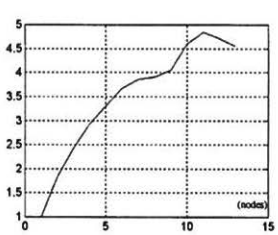

Figure 5.2 Speedup for 121-10-121.

In order to perform performance comparisons, the third principal component was extracted by means of sequential and parallel implementations in the TN-310 system. The unit parallelism approach used 3 layers of parallelism. The sequential implementation required approximately 33.6 minutes in the TN-310 system and the parallel implementation required 14.5 minutes, which represents a speedup factor of 2.32 and an efficiency of 77%.

6 Conclusions and Further Work

As a case study, a neural high-energy particle discriminator based on principal component analysis was designed. Using the information of a fine-grain energy measurement detector, a calorimeter, seven components were extracted from original data vectors with 121 components. In this application, principal component extraction and final particle discrimination are both performed by neural networks. For performance evaluation, the extraction of the third principal component showed that the parallel approach using unit parallelism was capable to reduce by a factor of 2.3 the required processing time.

For the near future, the implementations of neural networks in the TN-310 system will integrate more preprocessing techniques (topological mapping, discriminating and relevance analysis, mapped principal component extraction, etc) and the user interface for defining the network parameters and providing tools for the analysis of system performance will be improved to accommodate a flexible and user-friend development environment. To improve processing speed, an intensive use of the DSP co-processors will be pursued. This may be achieved by using the two memory buffers, which are shared by the transputers and DSP[12]. Also, in order to explore the very good matching that exists between DSP architectures and neural processing requirements, a similar development environment will be built for powerful multiprocessor DSP based boards, which are presently available in the market at much more reasonable prices. These studies are under development.

Acknowledgement

We are thankful to the financial support that has been provided by CNPq, FUJB (Brazil), CEADEN (Cuba) and the European Commission to this work. We also thank the RD1 and Trigger/DAQ collaborations at CERN for providing the data samples and for fruitful discussions concerning this work.

References

[1] E. Littmann: Trends in Neural Network Research and an Application to Computer Vision. New Computing Techiniques in Physics Research III, World Scientific, pp 253-268, 1994.

[2] S. Haykin: Neural Networks, a Comprehensive Foundation. Macmillan, 1994.

[3] Telmat Multinode. TN-310 System Manual and Training Set. France 1995.

[4] The SGS Thomson. T9000 Transputer Hardware Reference Manual. France, 1993.

[5] Edited by M.D. May, P.W. Thompson, P.H. Welch. Networks, Routers & Transputers: Function, Performance and Application. IOS Press, Netherlands, 1993.

[6] . J. Hertz, A. Krogh, R.G. Palmer: Introduction to the Theory of Neural Computation. Addison-Wesley, 1991.

[7] T. Nordström, B. Svensson: Using And Design Massively Parallel Computers for Artificial Neural Networks, Journal Of Parallel And Distributed Computing 14, pp. 260-285, 1992.

[8] M. Besh, H.W. Pohl: PROMOTER- Application Study. How to Simulate Artificial Neural Network on Large Scale Parallel Computers Exploiting Data Parallelism and Object-Orientation, Technical Report TR-94020. Nov. 1994.

[9] L. Coetzee: Parallel Approaches to Training Feedforward Neural Nets. PHD Thesis. Univ. of Pretoria. Feb. 1996.

[10] Rabello dos Santos. Projeto Final No. 38. Sistema de Clasificação de uma Maquina com Processamento Distribuido. DEL. UFRJ.1997.

[11] J.M. Seixas, L.P. Caloba, B. Kastrup: A Neural Second-Level Trigger System Based on Calorimetry and Principal Component Analysis. New Computing Techniques in Physics Research IV, pp 545-550, World Scientific, 1995.

[12] Telmat Multinode. DSP HTRAM Software. User Manual, France , 1995.