NeuSim: A Modular Neural Networks Simulator for Beowulf Clusters

Carlos J. García Orellana, Ramón Gallardo Caballero, Horacio M. González Velasco, and Francisco J. López Aligué

Dpto. de Electrónica. Facultad de Ciencias
Universidad de Extremadura
Avd. de Elvas, s/n. 06071 Badajoz - SPAIN
carlos@nernet.unex.es

Abstract We present a neural networks simulator that we have called NeuSim, designed to work in Beowulf clusters. We have been using this simulator during the last years over several architectures and soon we want to distribute it under GPL license. In this work we offer a detailed description of the simulator, as well as the performance results obtained with a Beowulf cluster of 24 nodes.

1 Introduction

As we know, the simulation of neural nets is an expensive work from the computational point of view. In the last years, the growing speed and capacity of the computers have allowed the simulation of neural networks with few neurons to be done in any standard computer, without the necessity of using workstations nor specific hardware which was so common some years ago [1,2].

However, when the objective is to work with large size neural networks and/or with biologically accurate models, we find that the simulation time and necessary memory increase a lot, the simulation in a conventional computer becomes very difficult. Some authors [13] work with simplified models for which they obtain results similar to the most accurate simulations. How ever, this is not always possible. When we need to do these simulations, parallel simulators running on a network of workstations (NOW) or on a cluster are very useful [9,11].

Our work is centered on the simulation of neural nets under Beowulf clusters [3,4]. A cluster is a collection of computers connected by a fast interconnection system. In particular, a Beowulf cluster is characterized to be built using commodity hardware (although not always). Usually, processors of the x86 family, normal memory of PC’s, standard cases and fast-ethernet network adapters are used. Sometimes, hard disks are used in each node, but it is not always necessary. In general, one of the nodes acts of supervisor (master) and the other ones carry out the work (slaves).

The connection among nodes and the election of the processors are two of the critical points in the cluster design. Although it is habitual to use switched fast-ethernet (with one or several cards for node), there are other solutions that
have better performance like Myrinet [10] or Gigabit ethernet, however, the cost is much bigger.

With respect to the number and type of processors, it depends on the application. In general, when the applications make intensive use of the memory it is advisable to use monoprocessors. However, in another type of applications it can be better to use SMP multiprocessors (symmetric multiprocessing). Regarding the type, if we limit ourselves to compatible x86 processors, it seems that Athlon AMD processors are a better option instead of Pentium III, not only because of cost, but also because of their best performance in floating point calculations. There is even comparatives in which the Athlon overcomes to the Pentium 4 in floating point operations [12], in spite of the excellent bandwidth with the main memory of Pentium 4.

Another fundamental component of a cluster is the operating system to install. In a Beowulf cluster the operating system is usually Linux. As it is known, Linux is a freeware version of UNIX and it is occupying a more and more important position in the world of operating systems. Without doubts, the administration of the cluster is one of the most complicated tasks, though there are Linux distributions adapted for clusters that facilitate the installation of the nodes, offer nodes supervision, common space of processes, etc. [8].

In the following section we will describe the structure and characteristic of our simulator. After this, we take into account the simulator performance, and later we will show the results, both in time of simulation and in degradation of the parallelism (efficiency), of our system with several sizes of neural nets. We will conclude with the conclusions of our work.

2 Description of the simulator

In this section, we are going to describe to the types of neuronal networks that we want to simulate and the structure of the simulator.

2.1 Neural Network models

As we have commented in the introduction, our objective is that the simulator can work with neural networks in general, but with preference neural nets of large size and with algorithms computationally expensive.

During the development of the simulator, we have carried out a modeling to facilitate the simulation of neural networks [7]. An updated object modeling diagram can be observed in the Figure 1.

From the objects that are observed in that figure, we would like to stand out the classes CONEX, CX, ACT_CONTROL and NN_SCHED.

CONEX indicates us that a connection exists between two layers, without specifying the exact form of this connection, that work is left to the objects of the class CX (connection pattern). To the class CONEX also goes bound the learning algorithm, which can be implemented by an loadable independent module.