

Comparing the performance of fifteen processors using the CPS unit

A. Pinti, J.-C. Gross  tie

*European Commission, Joint Research Centre, Institute for
Systems Engineering and Informatics - TP 361, 21020 Ispra, Italy*

Abstract

Calculation performances of processors are essential for reducing calculation time of connectionist networks. In this study, 15 processors were evaluated and compared by using a neuronal power test. The power test is completely described and specified so it can be easily implemented on any type of machine. Test results show that today, it is not necessary to use supercalculators anymore to be able to use artificial neuron networks. They also show that within a ten year span, calculation power of processors has increased by a factor 50. Evaluation of performances in units of Connection Per Second (CPS) allows estimation of future performance of a connectionist application by knowing only the number of connections of the neuron network and the type of processor used.

1 Introduction

The main reason for comparing calculation performance of processors available on commercial systems is to be able to chose the processor according to one's needs. Presently, processor architecture is either of the Reduced Instruction Set Computer (RISC) type, the Complex Instruction Set Computer (CISC) type, the Digital Signal Processor (DSP) type, or the vector type. Moreover, some have

90 High-Performance Computing in Engineering

pipelines or superscalar calculation units. These different parameters greatly influence calculation performance. To portray machine performance, constructors will indicate peak performance in Millions of Instructions Per Second (MIPS) and in Millions of Floating-point Operations Per Second (MFLOPS). These performances are practically impossible to obtain with a real problem. It turns out that it is necessary to construct comparative tests adapted to the problem to be processed. Although many benchmarks have been developed by researchers, there is no perfect benchmark.

Research groups are trying to design benchmarks that are operational for all types of applications. To cite a few: Perfect benchmark [1] and Euroben [2] which group combinations of several scientific applications.

The first published benchmark ever developed was the Whetstone [3]. It was written in 1976 in Algol 60 language. The unit of measurement of this benchmark is the Kilo Whetstone Instruction Per Second (KWIPS). This program contains an important amount of data in real values and operations in floating point mode. It also has an important number of mathematical function calculations, which are not the main characteristic of current scientific applications. An improvement of it was the Dhrystone benchmark. The benchmark usually found in the literature is the System Performance Evaluation Cooperative (SPEC) [4] test. It can be evaluated in floating point mode with SPECfp92 (14 applications) or else in integer mode with SPECint92 (6 applications). This benchmark has more than 150000 program lines written in high level language and because of its size it is used on mini-computers and large systems. For these systems there are also the LINear algebra PACKage (LINPACK), the Los Alamos National Laboratory (LANL), Numerical Aerodynamic Simulation (NAS), Livermore, EDN, Rhealstone, Byte and CPU2 tests, most having been developed in FORTRAN but also in C [5, 6, 7, 8]. To test performances of personal computers (PC), the most popular test is the Sieve of Eratosthenes [9] benchmark.

The differences between these benchmarks are the tasks to be executed, the number of tasks, the number of calls to the processor's elementary routines (which can be cabled), the size of the program and the programming language. In some cases, the performance of access (input/output) to peripherals or to outside systems through a communication network (i.e. Ethernet, ATM, Token Ring) is important when the amount of shared data to be transferred is large. This has been demonstrated in particular for parallel or distributed systems. In other cases, the speed of these peripherals (i.e. disk, screen), of buses (i.e. SCSI, PCI) is essential when the amount of local data to exchange is

High-Performance Computing in Engineering 91

considerable (transfer of a database or of images). The intrinsic speed of the processor is essential for applications in numerical calculations. In artificial intelligence, the calculation mode and the precision of parameters are important in order to calculate the number of neuron networks. The required calculation power depends on the number of connections in the network. Thus it appears interesting to develop a benchmark that is easily portable in order to compare performances of processors using the number of Connections per Second (CPS) unit of measurement.

The article is organized in the following manner: after describing the main benchmark algorithms as well as their objectives, section 2 describes the neuronal benchmark developed to test processor power in CPS units. The neuronal network architecture, the synaptic configuration and data presented to the network are described in detail. Section 3 gives the results on power obtained on 15 processors amongst which are RISC, CISC and vectorial architecture processors. Section 4 concludes about the different results obtained with the developed benchmark and gives future developments and applications of the test. Finally section 5 indicates the references used to realize this study.

2 Benchmark description

With the objective of designing an easily portable benchmark, the Multi-Layer Perceptron type neuron network was used to compare the power of processors. It is the most popular network in connectionist applications. It is described in all neuronal calculation works of reference [10, 11, 12]. It is executed in operational phase. In the training phase, weights are updated with a back-propagation of errors according to the gradient method. If it had been executed in this phase, the algorithm would have been more complex and the unit of measurement would be modified in units defined by the number of Connections and Updates of weights Per Second (CUPS).

The following figure illustrates the general characteristics of the artificial neuron network selected for the benchmark.

92 High-Performance Computing in Engineering

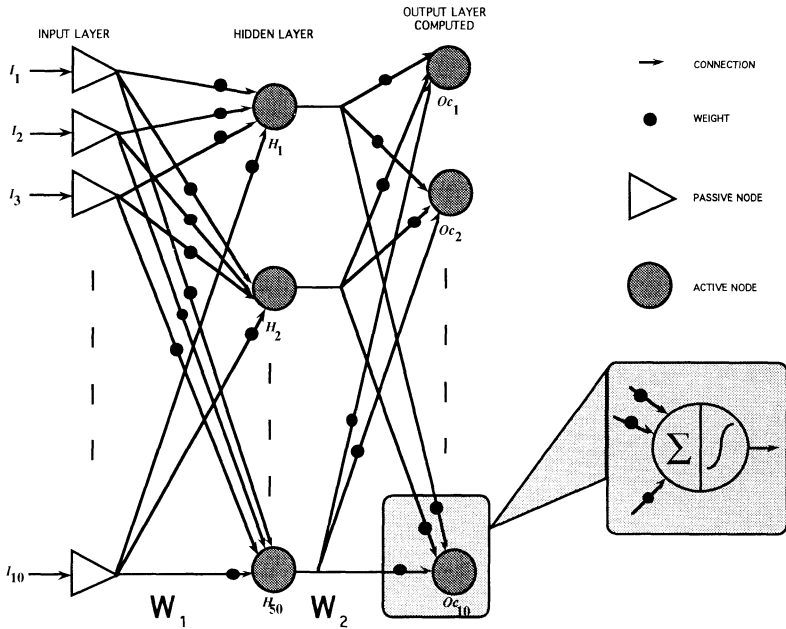


Figure 1: Structure of multi-layer Perceptron benchmark

The standard architecture of the artificial Multi-Layer Perceptron (MLP) neuron network is defined by parallel calculation elements arranged in layers. All the neurons of one layer are connected to all neurons of the following layer. Practically, this model of neuron network is used with one or two hidden layers. Pattern recognition first used this model [13]. In the MLP neuron network with 3 layers, shown in figure 1, there is an input layer I with inactive neurons. This layer does not do any calculation, but it receives the input signal. A hidden layer H and an output layer O each contain active neurons. The output layer is called the decision layer and the hidden layer is called the association layer. Information propagates from the input to the output layer. Each neuron has an output and several inputs. The inputs of active neurons are weighted by synaptic weights. For the benchmark network, there are 2 matrices of synaptic weights W_1 and W_2 . The W_1 matrix connects the input layer I to the hidden layer H. The W_2 matrix connects the hidden layer H to the output layer O. The network has 10 neurons in the input layer, 50 neurons in the hidden layer and 10 neurons in the output layer. That is a total of 60 active neurons and 10 inactive neurons. In this configuration, the total number of connections is 1000 distributed among the 3 layers. The distribution is the following: 500

High-Performance Computing in Engineering 93

connections (10×50) between the input layer and the hidden layer and 500 connections (50×10) between the hidden layer and the output layer.

A linear neuron modeled by a continuous automaton is the elementary active neuron used in this study. Its inputs and its output have real values, the input function is linear, the activation function is sigmoidal. The value θ represents the threshold of the linear neuron. The sum of the inputs of the neurons of the hidden layer H is calculated with the following expression:

$$u^{Hk} = \sum_{i=1}^{10} W_{li}^{Hk} I_i - \theta \quad (1)$$

with $I \in \mathbb{R}^{10}$, $W_1 \in M_{10,50}(\mathbb{R})$ and $\theta = 1.0$.

the sum of the inputs of the neurons of the output layer is calculated in the same way:

$$u^{Ol} = \sum_{j=1}^{50} W_{2j}^{Ol} H_j - \theta \quad (2)$$

with $H \in \mathbb{R}^{50}$, $O_c \in \mathbb{R}^{10}$, $W_2 \in M_{50,10}(\mathbb{R})$ and $\theta = 1.0$.

The output of each neuron is calculated as following:

$$F(u) = \frac{1 - e^{-2\gamma u}}{1 + e^{-2\gamma u}} \quad (3)$$

with $\gamma = \frac{1}{2}$.

In order to obtain a representative test of applications using connectionist networks for pattern recognition [14, 15, 16], the configuration of the test was defined with the following parameters: 10^4 input vectors, 10^3 synaptic weights. The ten thousand (10^4) I input vectors are initialized with the same line vector of dimension 10, of which all components are equal to one:

$$I = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]'$$



94 High-Performance Computing in Engineering

Values belonging to the matrices W_1 and W_2 are also initialized with unit value synaptic weights. The memory occupied by the synaptic weights is negligible. It is 4000 bytes (1000×4) for calculations in single precision floating point mode (4 bytes per value). In the same way, active memory occupied by the 10^4 I vectors to be propagated across the MLP neuron network is about 390 Kbytes ($10000 \times 10 \times 4$). The live memory space occupied by these different values is equal to 394 Kbytes.

The number of elementary operations and of mathematical functions executed during execution of the program is shown in the following table:

Operation type	Number
multiplication	1000
addition	940
subtraction	60
transfer function (TH)	60

Table 1 : Number of operation types

The program was developed in C ANSI programming language so that it could easily be ported between computers. Indeed, all present systems have at least one C compiler of the cc or gnu type. The program uses the "math.h" mathematical library. The intensive procedure for calculation contains 50 instruction lines contained in 2 loops. The two loops correspond to propagation of data through the layers.

Practical benchmark algorithm :

- Step 1 - Starting time
- Step 2 - Initialize weights W_1 , W_2 and the 10^4 I input vectors
- Step 3 - Input the first I vector to the network
- Step 4 - Calculate activity of the 50 H layer neurons according to (1) and (3)
- Step 5 - Calculate activity of the 10 O layer neurons according to (2) and (3)
- Step 6 - Until all I vectors have been propagated, input the next I vector to the input layer and return to step 4
- Step 7 - Ending time
- Step 8 - $CPS = (Ending\ time - Starting\ time) / 10^3$
- Step 9 -End of benchmark

3 Benchmarking 15 processors

The benchmark was executed for 15 processors of RISC, CISC and vectorial types with the Unix operating system. The program was compiled with the C compiler of the Unix system (cc). The compilation optimization option (-O) was used in order to obtain best performance of the processor without any modification of the program. The automatic vectoring options were used on vectorial processors. Calculations were done in single precision 32 bit floating point mode even though some processors have calculation units working with 64 bits. The results of the benchmark are shown by order of increasing processor power in the following table:

Processor	Clock Mhz	CPS
T800	20	2.7
80486/Dx	25	4.5
SPARC1	25	9.5
T9000	20	11
SPARC2	25	21
68040	40	22
SPARC20	50	66
88110	50	74
I860	40	90
RS6000	33	100
R4000	100	111
ALPHA AXP	150	166
CONVEX	166	166
YMP8	166	294
YMP/C90	250	500

Table 2: The CPS performance for various processors

4 Conclusions and future work

This work has resulted in the development of a neuronal benchmark to compare performances of 15 processors having different internal calculation architecture. The benchmark has been named *Neurobench*. Performance results are given in CPS units. The advantage of this unit is that it predicts future performance of a processor used within the context of connectionist applications.

96 High-Performance Computing in Engineering

The best calculation performance among the 15 compared processors were obtained from the processor of the CRAY YMP/C90. Benchmark results showed that the number of CPS obtained with the Alpha-150 is only 3 times smaller than that obtained with the CRAY. Today it is thus possible to obtain with a high-level PC a calculation power equivalent to that of a CRAY supercomputer (under the conditions of the test which do not favor the CRAY). The mediocre results obtained with the CRAY are due to one parameter. The size of an optimal vector for a CRAY YMP/C90 is 256 and the length of the mean vector obtained with the *Neurobench* test on the CRAY was 12. Performance can therefore be multiplied by 21 in the best of cases, which would bring the CRAY performance to 10500 CPS. One can also note that within one decade, RISC processors (T800-R4000) have multiplied their performance power by 50.

Many factors influence processor performance and it is always possible to optimize a processor program. The performances obtained in this study can be improved for each processor by modifying the program text, and this is particularly true for vectorial processors (CRAY, CONVEX). Optimizing compilers have been very important for increasing CISC processor performance, and with the second generation of processors which execute several operations at the same time (superscalar techniques), the importance of these compilers is even more obvious. This is also true for processors having an instruction pipeline. Depending on the number of stages of the pipeline, there may be more or less sequence ruptures in a program. Also, the program text must be modified in order to improve calculation time. It is also possible to increase performance by using optimized calculation libraries and compilers other than cc or as well to rewrite the program in a low level language (assembly).

The objective of this study was not to obtain the best performance but indicative performances with a benchmark which is easily portable from system to system.

The benchmark done in this study in single precision floating mode can also be done in another calculation mode. Processors that have a 64 bit floating point calculation unit give better results in double precision floating point calculation mode. To do the test in 8, 16, 32, or 64 bits integer calculation mode, the transfer function would have to be changed. This function could be for example a simple sign function. It is also possible to include the configuration of the MLP neuron network as a parameter of the test: *Neurobench* (10*50*10) just as for the test Linpack (100*100). Since the memory size used modifies performance, it is important to indicate the parameters of *Neurobench*. It would

High-Performance Computing in Engineering 97

be possible to make an extension of the test with a *Neurobench* configuration applied to image processing.

Since the neuron network intrinsically has a high level of parallelism, it would also be interesting to develop this performance test on parallel machines belonging to the three large families Single Instruction Multiple Data (SIMD), Multiple Instruction Multiple Data (MIMD) and Single Program Multiple Data (SPMD). For parallel machines, organization of communications and access to shared data are additional problems.

References

1. Berry M., *The perfect club benchmarks : Effective Performance Evaluation of Supercomputer*, International Journal of Supercomputer Applications, n. 3, pp. 5-40, 1989.
2. Van der steen A. J., *The benchmark of the Euroben Group*, Parallel computing, vol. 17, n. 10&11, 1991.
3. Curnow H. J. & Wichmann B. A., *A synthetic benchmark*, The computer journal, 19, 1, pp. 43-49, 1976.
4. Uniejewski J., *SPEC benchmark suite : designed for today's advanced systems*, SPEC newsletter, n. 1, 1989.
5. Reinhold P. W., *A detailed look at some popular benchmarks*, Parallel Computing, n.17, pp. 1153-1172, 1991.
6. Weicker R. P. ., *An overview of common benchmarks*, IEEE computer, pp. 65-76, December 1990.
7. Berry M., Cybenko G. and Larson J., *Scientific benchmark characterizations*, Parallel computing, n. 17, pp. 1173-1194, 1991.
8. Price W. J., *A benchmark tutorial*, IEEE micro, pp. 28-43, October 1989.
9. Gilbreath J. G., *Erastosthenes revised*, Byte, pp. 283-326, January 1983.

98 High-Performance Computing in Engineering

10. Freeman J. A., Skapura D. M., *Neural networks algorithms applications and programming techniques*, Addison Wesley, 1991.
11. McCord Nelson M. and Illingworth W. T., *A practical guide to neural nets*, Ed. Addison Wesley, 1991.
12. Davalo E. and Naim P., *Des réseaux de neurones*, Ed. Eyrolles, 1990.
13. Pao Y.-H., *Adaptive pattern recognition and neural networks*, Ed. Addison Wesley, 1989.
14. Yao Y., Freeman W. J., Burke B. and Yang Q., *Pattern recognition by a distributed neural network : an industrial application*, Neural Networks, vol. 4, pp. 103-121, 1991.
15. Hoffman J., Skrzypek and Vidal J. J., *Cluster network for recognition of handwriten cursive script characters*, Neural Networks, vol. 6, pp. 69-78, 1993.
16. Battiti R., Colla A. M., *Democracy in neural nets voting schemes for classification*, Neural Networks, Vol. 7, pp. 691-707, 1994.

Acknowledgments

This work was supported by the European Communities Programme, Human Capital and Mobility programme fellowship at Joint Research Centre of Ispra (Italy).