Journal of Mathematical Modelling and Algorithms **1:** 215–223, 2002.
© 2002 *Kluwer Academic Publishers. Printed in the Netherlands.*

A Multicomputer System Performance Predictor Based on an ALN Neurochip

O. B. EFREMIDES[1] and M. P. BEKAKOS[2]
[1]*Department of Informatics, Athens University of Economics and Business, Athens 10434, Greece. e-mail: obe@aueb.gr*
[2]*Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, 67100 Xanthi, Greece. e-mail: mbekakos@ee.duth.gr*

(Received: 25 September 2001)

Abstract. Although a large number of performance evaluation tools are available today, the efficient performance prediction of a multicomputer system remains a difficult task. In this work, the development of a system capable of predicting the timings for an application executed on the multicomputer Parsytec GCel3/512 is investigated.

Mathematics Subject Classifications (2000): 68H14, 68H20, 68H10, 68Q85.

Key words: performance, multicomputer, predictor, ALN, neurochip.

1. Introduction

The computer systems performance evaluation and prediction has always concerned the scientific community [2, 4, 6]. Despite a rapid increase in processor speed, due to the improvements in microprocessor technology, the overall application performance is still drastically affected by the communication capabilities of the supercomputing complex in hand. The performance prediction of an application to be executed on a system under certain communication requirements and the ability of the system structure to comply with these requirements, relating to the corresponding computational load, are essential.

Herein, an adaptive neural network [1] is built, trained to predict the timing results for an application executed on the multicomputer system Parsytec GCel3/512. The training and the testing data set for the learning phase of the network have resulted from the performance evaluation of the real system using a parametric synthetic workload produced through a distributed-memory procedure [5].

After the completion of the training phase, a prediction function is available. Further, the layout of the prediction system is proposed and a software tool implementing this function is developed. The entire prediction function is based on a set of *five* different parameters [3]: The *number of processors*, the *average amount of the point-to-point communication* for the data transfer, the *degree of distribution* of the elements amongst the nodes, the *computation-communication ratio* and the *size of the problem* investigated each time.

2. Forming the Data Set

A parametric synthetic workload produced using a Sparse Matrix Dense Vector multiplication algorithm, $[\Pi_\alpha \times \Delta_\pi]$, given by Equation (1)

$$\mathbf{y} = A \times \mathbf{x}, \tag{1}$$

where $\mathbf{y}$ is the vector of the results and A, $\mathbf{x}$ are the sparse matrix and the dense vector, respectively. More specifically, the four basic parameters concerning the production of the workload are defined as follows [3]:

DEFINITION 1. As the number of processors, P, is defined, the total number of independent nodes which are involved in the computations and are interconnected through the communication network.

DEFINITION 2. As the average number of point-to-point communications, U, for the passing of the information elements, is defined the number of the elements of vector $\mathbf{x}$ involved in the computations executed by a processor, but not belonging to this processor, i.e., the number of the elements of vector $\mathbf{x}$ required by a processor from another processor.

DEFINITION 3. As the distribution degree, D_i, is defined, the number of processors which require an element of vector $\mathbf{x}$ not belonging to them. Further, as the average of the distribution degree, D, is defined the average of the nonzero $D_i's$. Thus, $1 \leqslant D \leqslant (P+1)$.

DEFINITION 4. As the computation-communication workload ratio, *CCR*, is defined the ratio of the floating point multiplication-addition per processor over the number of point-to-point communications for data transfer per processor.

2.1. THE PARSYTEC GCEL3/512 SYSTEM

The Parsytec GCel3/512 multicomputer system consists of 512 Inmos Transputer T805 processors organized in gigacubes (i.e., mesh of 64 processors). It achieves a maximum performance of 22 GFLOPS and it has a total memory of 2 Gbytes. The topological organization of the system is *mesh*. The T805 processor is a 32-bit RISC processor with the following characteristics: Operation frequency: 30 MHz, floating point unit: 64 bit, maximum performance: 4.3 MFLOPS, cache memory: 4 Kbytes (on chip), external memory DRAM: 4 Mbytes.

A number of processors is exclusively allocated to the user. Hence, the user executes the application without any interfering with other user applications. The system runs under the PARIX (PARallel extensions to uniIX) operating system. The PARIX supports parallel programming via special routines, provides an integrated programming environment and program execution management at the user and the administrator level, etc.

2.2. PERFORMANCE RESULTS

As was mentioned in the previous section, the required number of processors is exclusively allocated to the user application. Two different gigacubes were set in order to describe how the performance evaluation of the system is carried out using the synthetic workload and how the achieved results can be used to compare this system with others. First, the program was executed on two different gigacubes and then the entire system (512 processors) was set in order to describe how the performance evaluation of the system is carried out using the synthetic workload. The algorithm is implemented using Fortran 77 programming language. The experimentation sequence (i.e., 40 different executions for each different set of parameters, for statistical accuracy reasons) calculates the execution time of the algorithm $[\Pi_\alpha \times \Delta_\pi]$ under different values for the communication free computations, (X), the number of point-to-point communications $(U = M)$, the distribution degree of the vector $\mathbf{x}$ elements D, and, finally, the number of processors (P) [3].

2.2.1. *Variations of Communication-Free Computations*

To measure the influence on the performance when the communication free computations X vary, a number of different synthetic workloads was timed with the size of the matrix A $(N = S \times P)$ being constant. The parameter X takes different values, $X = 500, 1000, 1500, 2000, 2500, 3000, 3500$. In this case, there is no need for internode communication, hence $U = M = 0$. The program is executed for each different parameter set on a varying number of processors ($P = 2, 4$, 8, 16, 32, 64), in each of the two gigacubes. For the evaluation of the entire system the size of the matrix A was set constant ($S = 35$). The parameter X takes different values, $X = 200, 400, 600, 800$. Again $U = M = 0$ and the number of processors were ($P = 32, 64, 128, 256, 512$).

2.2.2. *Variations of Point-to-Point Communications*

The performance has been measured for varying point-to-point communications with a constant matrix size ($S = 100$) and various combinations of the number of processors parameter ($P = 2, 4, 8, 16, 32, 64$) in each of the two gigacubes. When $U = M$ the total workload (computation and communication) is determined as $(U + X) = 200$ and $(U + X) = 2500$ (the parameter U takes, in turn, the values $U = 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50$). The entire system was evaluated with a constant matrix size ($S = 35$) and various combinations of processors ($P = 32$, 64, 128, 256, 512). Again, for $U = M$ the total workload was determined as $(U + X) = 200$ and $(U + X) = 800$ (the parameter U takes, in turn, the values $U = 0, 5, 10, 15$). Since $U = M$ the distribution degree D is constant and equal to 1, thus, each element of vector x is required only by one processor (not the owner) in order to complete the calculations. This implies that each processor knows that there is no need for multicasting.

2.2.3. *Variations of the Number of Processors*

The variation of the timing results is studied when the number of processors vary and the number of multicasting ($M = 50$ for each gigacube) was kept constant. The combinations of processors are ($P = 2, 4, 8, 16, 32, 64$) and the constant size of matrix used ($S = 100$). The parameter U takes the values $U = 0, 50, 130, 200, 300, 400, 500$ in different executions and the number of communication free computations is $X = 1000 - U$. For the evaluation of the entire system the combinations of processors are ($P = 32, 64, 128, 256, 512$) and the constant size of matrix used ($S = 35$). The parameter U takes the values $U = 0, 5, 10, 13, 20, 30, 40, 50$ in different executions and the number of communication-free computations is $X = 120 - U$.

2.2.4. *Variations of the Elements' Distribution Degrees*

The size of the matrix remains constant, $S = 100$, and the combinations of processors are ($P = 2, 4, 8, 16, 32, 64$) in each of the two gigacubes. The number of multicasting is constant ($M = 50$), while the total workload of computation and communication is $(U + X) = 1000$ and $(U + X) = 3000$ (the parameter U takes, in turn, the values $U = 0, 100, 200, 300, 400, 500, 600, 700, 800$). For the evaluation of the entire system the size of the matrix remains constant, $S = 35$, and the combinations of processors are ($P = 32, 64, 128, 256, 512$). The number of multicasting is constant ($M = 5$), while the total workload of computation and communication is $(U + X) = 120$ and $(U + X) = 360$ (the parameter U takes, in turn, the values $U = 0, 10, 20, 30, 40, 50, 60, 70, 80$).

3. An Integrated Predictor

The development of the proposed prediction system integrates hardware and software components. The layout of the entire system is presented in Figure 1.

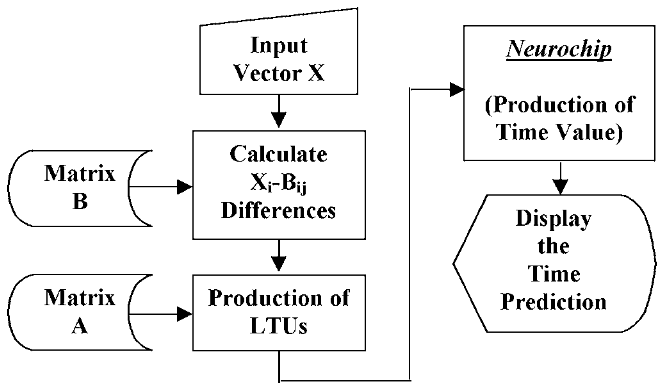
Input Vector X
Calculate X_i-B_{ij} Differences
Matrix B
Production of LTUs
Matrix A
Neurochip
(Production of Time Value)
Display the Time Prediction

Figure 1. Layout of the predictor.

Initially, using the real data set, an *Adaptive Learning Network* (ALN) is trained to produce the prediction function. An ALN is a type of an artificial neural system which uses only simple logical functions, such as AND, OR and NOT for building a decision tree architecture during the training. The knowledge is stored in the architecture of the network and in the functions of the system nodes.

The basic components of an ALN are: (i) *the Linear Threshold Units* (LTUs), which are the leaves of the decision tree architecture and (ii) other nodes which are AND and OR operators with two or more inputs [1]. Since the ALN is trained, several linearform equations (in our case 890) are produced. The form of these equations is given by Equation (2):

$$LTU_i = \sum_{i=1}^{n} a_{ij}(x_i - b_{ij}), \tag{2}$$

where coefficients a_{ij}, b_{ij} are estimated during the training phase for each input measurement with confidence interval 90% (error $[-0.668685, 0.563254]$). The resulted *root mean square error* is 1.25437 over 854 training points (set of parameters).The prediction function consists of eight different blocks of LTUs combinations. The form of these blocks is described by relation (3).

$$\begin{aligned} &MAX(MIN(MAX(84, 85), 86, MAX(MIN(67, 68), 69, 70),\\ &\quad MAX(MIN(71, 72), 73, 74), MAX(64, 65, 66),\\ &\quad MAX(75, 76, 77, 78), MAX(MIN(55, 56), \cdots\cdots); \end{aligned} \tag{3}$$

When new data is entered into the system, the LTUs values are produced. Then these values are fed into the neurochip. The system executes the appropriate block according to the decision tree depicted in Figure 2 and the predicted time result is calculated. Note that, the numbers in the brackets of the classification function correspond to the LTUs.

4. The Neurochip Design

As it is determined by the prediction function, the hardware implementation of the neurochip involves two different subcircuits named MIN and MAX, respectively. Each subcircuit accepts two 32-bit numbers. The MIN/MAX subcircuits output the smaller/greater of the two input numbers, respectively. These subcircuits can be designed using random logic and structured techniques. With the random logic technique primitive gates are used, such as AND, OR, NOT, etc., while with the structured design approach existing circuits are used, such as comparators, multiplexers, etc.

```
0 : (x2 <= 9.8733084914990904e+305) ? 1 : 2;
1 : (x0 <= 1.1235582092889473e+307) ? 3 : 4;
2 : block 1;
3 : (x0 <= −2.457783582819572e+307) ? 5 : 6;
4 : (x0 <= 3.49356380700782e+307) ? 11 : 12;
5 : (x2 <= −1.0662999742778974e+307) ? 7 : 8;
6 : (x1 <= −5.0560119418002625e+307) ? 9 : 10;
7 : block 0;
8 : block 4;
9 : block 3;
10 : block 5;
11 : (x2 <= −1.0309959421811433e+307) ? 13 : 14;
12 : block 6;
13 : block 2;
14 : block 7;
```

Figure 2. Decision tree for the block execution.

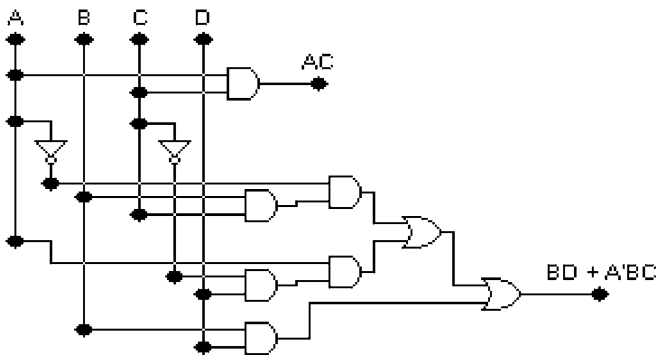

Figure 3. Circuit for *MIN* function using random logic.

4.1. RANDOM LOGIC APPROACH

A truth table is built through the typical procedure for both MIN and MAX operations. The logic function of each table is simplified using the Quine–McCluskey technique. The logical diagrams of the circuits are developed using the resulting simplified functions. For example, a circuit capable of executing the MIN operation for two (AB, CD) 2-bit numbers is given in Figure 3.

4.2. COMBINATORIAL LOGIC APPROACH

In the second approach, already fabricated circuits are also considered. Each subcircuit can be build using a 32-bit comparator and a 64 $\times$ 32 multiplexer with two *selection* lines. The comparator compares the two 32-bit numbers and produces a sequence of three bits for L(ess), G(reater), and E(qual). For example, an output of (100) denotes that the first of the two input numbers is less than the second. The formulae for a 32-bit comparator are described by Equation (4).

$$\begin{aligned}
E &= \overline{(x_{31} \oplus y_{31})} \cdot \overline{(x_{30} \oplus y_{30})} \cdots \overline{(x_1 \oplus y_1)} \cdot \overline{(x_0 \oplus y_0)},\\
G &= x_{31} \cdot \overline{y_{31}} + \overline{(x_{31} \oplus y_{31})} \cdot x_{30} \cdot \overline{y_{30}} + \overline{(x_{31} \oplus y_{31})} \cdot \overline{(x_{30} \oplus y_{30})} \cdot x_{29} \cdot \overline{y_{29}} + \\
&\quad + \overline{(x_{31} \oplus y_{31})} \cdot \overline{(x_{30} \oplus y_{30})} \cdot \overline{(x_{29} \oplus y_{29})} \cdots \overline{(x_1 \oplus y_1)} \cdot x_0 \cdot \overline{y_0}, \qquad (4)\\
L &= \overline{x_{31}} \cdot y_{31} + \overline{(x_{31} \oplus y_{31})} \cdot \overline{x_{30}} \cdot y_{30} + \overline{(x_{31} \oplus y_{31})} \cdot \overline{(x_{30} \oplus y_{30})} \cdot \overline{x_{29}} \cdot y_{29} + \\
&\quad + \overline{(x_{31} \oplus y_{31})} \cdot \overline{(x_{30} \oplus y_{30})} \cdot \overline{(x_{29} \oplus y_{29})} \cdots \overline{(x_1 \oplus y_1)} \cdot \overline{x_0} \cdot y_0.
\end{aligned}$$

The multiplexer has two selection lines. Thus, only the two bits resulting by the comparator are required in order to be decided which of the two input numbers will be produced by the circuit; namely the L output bit of the comparator is ignored. The truth table for the MIN subcircuit is given in Table I and its logic function is described by Equation (5):

$$f_i = \overline{B}x_i + \overline{A}By_i, \quad \text{for } i = 0, \ldots, 31. \qquad (5)$$

Similarly, for the MAX circuit, the logic function is described by Equation (6):

$$f_i = \overline{B}y_i + \overline{A}Bx_i, \quad \text{for } i = 0, \ldots, 31. \qquad (6)$$

To exemplify the discussed above, a 2-bit MIN subcircuit built using a 2-bit comparator and an 4×2 multiplexer is given in Figure 4. Again, the input numbers are symbolized as AB, CD, respectively.

5. The Simulator

The integrated performance predictor described in Figure 1 has been implemented using the Fortran programming language. In the program, 2D arrays have been used to store the coefficient matrices A, B and 1D array for the input vector.

The output of the program for an input vector ($P = 512, X = 240, U = 5, M = 80, A = 17920$) is presented in Figure 5. Note that, the execution complexity

Table I. Truth Table for MIN Operation

States	A	B	f_0	f_1	$\cdots$	f_{31}
L	0	0	x_0	x_1	$\cdots$	x_{31}
G	0	1	y_0	y_1	$\cdots$	y_{31}
E	1	0	x_0	x_1	$\cdots$	x_{31}

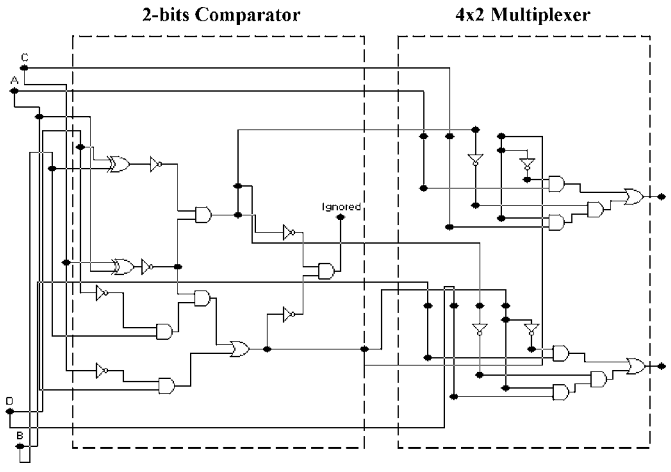

Figure 4. Circuit for *MIN* function using a comparator and a 4 × 2 multiplexer.

```
========================================
LOAD INPUT VECTORS X
LOAD VECTORS As
LOAD VECTORS Bs
CALCULATE Xi-Bi DIFERRENCES
LTU VALUES ARE PRODUCED
OUTPUT VALUE IS CALCULATED
========================================
PREDICTED TIME: 1669.01186695617600 sec
========================================
```

Figure 5. Output of the simulator.

in the Parsytec GCel3/512 multicomputer system of an application with the same set of parameters is 1669.800903 sec.

6. Conclusions

Herein, the development of an adaptive neural network, the logic design of a neurochip and the implementation of an integrated predictor, capable of predicting the time complexity of an application executed on the multicomputer system Parsytec GCel3/512, are proposed. The basic entity of the system is a prediction function which is produced using the adaptive learning network. Two different approaches for the digital design of the neurochip are discussed and the integrated system is implemented using the Fortran programming language.

References

1. Armstrong, W. W. and Thomas, M. M.: Adaptive logic networks, In: E. Fieseler and R. Beale (eds), *Handbook of Neural Computation*, Institute of Physics and Oxford Univ. Press, 1996.
2. Bard, Y.: The VM/370 performance predictor, *Computer Surveys* **10**(3) (1978), 333–342.
3. Boyd, E. L., Wellman, J.-D., Abraham, S. G. and Davidson, E. S.: Evaluating the communication performance of MPPs using synthetic sparse matrix multiplication workloads, In: *Proc. 1993 International Conference on Supercomputing*, 1993, pp. 240–250.
4. Boyse, J. W. and Warn, D. R.: A straightforward model for computer performance prediction, *Computer Surveys* **7**(2) (1975).
5. Efremides, O. B.: Distributed memory systems performance evaluation using synthetic parametric workloads, *J. Neural Parallel Sci. Comput.* (2000), 299–316.
6. Vrsalovic, D., Siewiorek, D., Segall, Z. and Gehringer, E.: Performance prediction and calibration for a class of multiprocessor systems, Rept. Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburg, PA, 1984.