TRAINING SET PARALLELISM IN PAHRA ARCHITECTURE

Liberios VOKOROKOS, Norbert ÁDÁM, Anton BALÁŽ
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, tel. 095/602 3175,
E-mail: liberios.vokorokos@tuke.sk, norbert.adam@tuke.sk, anton.balaz@tuke.sk

SUMMARY

Multilayered feed-forward neural networks trained with back-propagation algorithm are one of the most popular "on-line" artificial neural networks. These networks are showing strong inherit parallelism because of the influence of high number of simple computational elements. So it is natural to try to implement this kind of parallelism on parallel computer architecture. The Parallel Hybrid Ring Architecture (PAHRA), which is described in this article, provides flexible platform for simulation of multilayered feed-forward neural networks trained with back-propagation algorithm. The computational model of given architecture, bound to the modified error back-propagation algorithm, allows to describe the formal elements of parallel implementation of multilayered feed-forward neural network. It also allows the mathematical tool for verification of performance, which is used in simulation experiments of multilayered feed-forward network on specific hardware platform.

Keywords: *multilayered feed-forward neural network, error back-propagation algorithm, parallel computer architectures, PAHRA, processing element, computational model, processing time, simulation, experiment.*

1. INTRODUCTION

Artificial neural networks [2] are gaining high popularity over time in many application areas which emphasis is put on gathering of results in real time. Although there are many different implementations of artificial neural networks available for uniprocessor computer architectures based on the von Neumann type, most of these models require enormous amount of time for training and the live phase in case of large neural network (the number of neurons in network is ≥ 1000).

Therefore new concepts were developed, which contain the modification of original models and learning algorithms, together with the implementation of specialized [3] and / or universal [6] type of models based on the parallel computer architecture. These concepts focuses primary on the time reduction, especially on the time of learning of neural network. One of the most popular neural networks are multilayered feed-forward neural networks (FFNN) [2] with error back-propagation (BP) algorithm, which represent the most standard configuration of biological inspired mathematical models of simplified neural system. These networks represent massive parallel systems with a high number of simple process elements and therefore it is natural to try to implement this kind of systems on parallel computer architecture [9, 10].

2. THE PAHRA ARCHITECTURE

The *P*arallel *H*ybrid *R*ing *A*rchitecture (PAHRA) architecture is developed on DCI FEEI TU of Košice within the frame of projects [10] and [11] and is based on the conception of multiprocessor architectures. The PAHRA architecture is assigned primary on the implementation of FFNN with the use of training set parallelism. With the use of its architectonical conception and computational model it allows to separate the computation within neural network (processed on particular processing elements) from the hardware (the type of processing element).

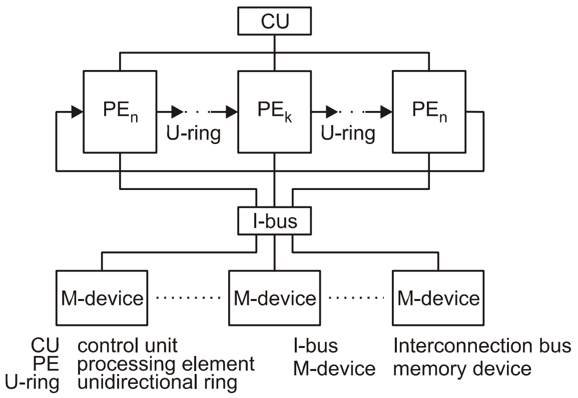


Fig. 1 The PAHRA architecture

The design of PAHRA parallel architecture (Fig. 1) was focused on the possibility to cower many classes of multiprocessor architectures as possible. Therefore the synchronizing interconnection bus was established as the central component of PAHRA architecture. The synchronizing interconnection bus allows to interconnect either independent computational elements (processors) or complex systems (computer clusters, MIMD systems, dataflow systems, etc.) [8, 9]. This conception of parallel architecture consists of defined number of processing elements (*n*) with the synchronizing simplex interconnection bus (type ring), control unit and interconnection bus which provides access to the memory (storage) devices.

Each processing element has its own execution unit (or execution units, in the case of complex systems located in node of synchronizing interconnection bus), local memory and set of communication lines to be able to communicate with other processing elements. The raw computational power of each processing element is characterized by the execution time of one elementary operation (arithmetical or logical). The synchronizing interconnection bus uses point-to-point method of connection with the synchronous communication.

The use of simplex communication and the overlay of computational and communicational phases disallow the all-to-all broadcasting which is characteristic for node parallelism (neuron and synapse parallelism) with vertical segmentation [7].

The time taken to inter-processor communication for the transmission of *m* words between adjacent processing elements is defined by

$$t_{com} = t_{com\_init} + m \cdot t_{word} \tag{1}$$

where t_{com} is the time taken to transmission of *m* words, t_{com_init} is the transmission initialization time and t_{word} is the time taken to transmission of one word. Within the frame of project [10] the task of mapping of FFNN into the PAHRA architecture was solved.

The mapping of neural network into a parallel environment is not a simple task. Two steps are needed to define the optimal mapping scheme. These steps contain the parallelization of classical BP algorithm (mentioned in [2]) and the identification of optimal decomposition of multilayered neural network (with the use of neural parallelism) or training set (with the use of training set parallelism). In next chapters, the mapping task related to the training set parallelism in PAHRA architecture is described.

2.1. Training set parallelism and PAHRA

When the training set parallelism as a form of course-grained data parallelism is applied, each processing element of the PHARA system gets a copy of a whole multilayered neural network and only the patterns are divided into processing elements (PE) [5]. In the suggested model, each PE$_i$ ($1 < i \le n$) accept subset of patterns P_i with the cardinality P^i, training set T with the cardinality P^{tot}, what is defined as follows:

$$\begin{aligned} &T = P_1 \cup P_2 \cup \ldots \cup P_n \\ &P_i \cap P_j = \varnothing : \forall i \ne j \qquad , \\ &P^{tot} = P^1 + P^2 + \ldots + P^n . \end{aligned} \tag{2}$$

Figure 2 shows the suggested conception of decomposition and allocation of training set on the PE's of the PAHRA architecture. Processing elements are executing forward and error back-propagation phase of BP algorithm with regard to the allocated patterns on each of them and they are computing the vector of weights and biases ($\mathbf{\Delta w}$). In the phase of weight update, each PE sends locally computed $\mathbf{\Delta w}$ (regard to the allocated set of patterns P_i) to its neighbor PE on the right side and receives the vector $\mathbf{\Delta w}$ from the PE on the left side.

The aim of applying the pattern parallelism during the learning phase of FFNN resulted into changes in the BP (mentioned in [2]) algorithm for the parallel execution in the PAHRA architecture.

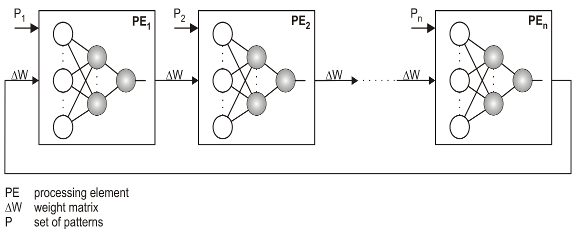


Fig. 2 The principle of decomposition and allocation of training set on processing elements

2.2. The model of computation and PAHRA

Steps of parallel execution of BP algorithm on ($1 \le i \le n$) in the PAHRA architecture are described as follows:

1. Each PE$_i$ initializes $\mathbf{\Delta w}$ for given epoch on zero (process $\mathsf{P}^i_{\Delta w}$ with the time $t^i_{\mathbf{\Delta w}_0}$).
2. Each PE executes forward and error BP phase. The deviation between the network output and the target value is computed also in this step (process P^i_{BP} with the time t^i_{BP}).
3. Processing elements swap $\mathbf{\Delta w}$ between each other. In the PAHRA architecture with the number of *n* processing elements, each PE requires to send (process P^i_{send} with the time t^i_{send}) and to receive (process P^i_{rec} with the time t^i_{rec}) ($n-1$) of $\mathbf{\Delta w}$. When the PE's are unable to finish the computation in the same time according to step 2 (e.g. in the case of heterogeneous behavior of PE's), latency occurs in data acceptance from the left (PE$_j$) neighbor processing element (latency $W^i_L \mathsf{P}_{rec,j}$) and in data dispatching to the right neighbor processing element (latency $W^i_L \mathsf{P}_{send,j}$). The update of all weights in the network (process P^i_w with the time t^i_w) on single PE with the index *i* can be started only after receiving of all partial $\mathbf{\Delta w}$ from the left neighbor PE's with the index *j*.
4. All processing elements update the weights and therefore gains new set of weights for their local copies of FFNN. Because each PE works above copies of the whole FFNN, the values of weights and biases will be the same.
5. All processing elements execute convergent test (the deviation between the network output and the target value is tested). If the process of learning is not converges, new iterating process is started from the step one. Otherwise the algorithm can be considered as finished.

The presented procedure represents basic steps for parallel BP algorithm with the learning-by-epoch (lbe) strategy. Figure 3 shows the meaning of processes, needed to realize steps 1 to 5, their interconnection and the mechanism of process synchronizing. By using a semiformal meaning, the graph shows the computational and communication processes, their parallel or concurrent execution during the execution of one iteration of designed parallel algorithm. On the basis of depicted graph it is possible to determine the realization time of individual processes which are participating on the process of learning based on the lbe strategy. Figure 3 shows the process synchronizing graph of parallel BP algorithm in PAHRA architecture with number of processing element $n = 3$.

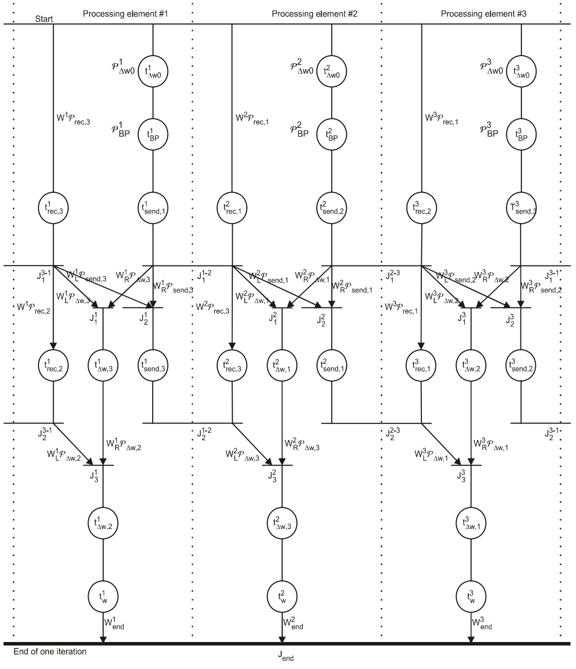


Fig. 3 Process synchronizing graph of parallel error back-propagation algorithm

Nodes in graph are representing processes and edges of latency between them. Time needed to process execution is listed in nodes.

The synchronization points between individual processes are represented by using places which are representing the conditions of concurrent processes execution (on individual levels *j*) in graph. Places in graph are given by using horizontal line which connects one or more processes above given line with one or more processes under given line. The meaning of places is as follows: process (processes) under the line can be started only after process (processes) above the line had finished.

There are two types of places. Intra-processor place (J_j^i) manages (synchronizes) processes runtime within one PE_i. Inter-processor place ($J_j^{i-(i+1)}$) manages (synchronizes) processes runtime on different PE_i. In this designated model, the functionality of every single computational and communicational process is described by using elementary operations from which they are consisting. The mathematical formulation of the price of computation depends on concrete implemental or simulation architecture.

2.3. The processing time

The computational model of parallel simulation of FFNN based on the principle of training set parallelism, was designed in [10] on the basis of ure 3. The functionality of processes is defined by using elementary computational and communicational operations which are defining given process. If we know the time of realization of elementary operations (which are defined in designed adaptation algorithm), it is possible to set up a mathematical formula for determining the price of realization of one epoch (or algorithm). The processing time is also defined as the sum of all prices of realization of individual processes and their latencies on their way from the "Start" point to the "End of one epoch" point. On the basis of this consideration and Fig. 3, the processing time in PAHRA architecture with the number of processing elements $n = 3$ (t_{epoch_3}) is defined as (3).

$$\begin{aligned} t_{epoch\_3} &= W^1\mathrm{P}_{rec,3} + t^1_{rec,3} + W^1\mathrm{P}_{rec,2} + \\ &+ t^1_{rec,2} + W^1_L\mathrm{P}_{\Delta w,2} + t^1_{\Delta w,2} + t^1_w + W^1_{end} + \\ &+ t^1_{\Delta w0} + t^1_{BP} + W^1_R\mathrm{P}_{send,3} + 2t^1_{send,1} + W^2_L\mathrm{P}_{\Delta w,3} \\ &+ t^2_{\Delta w,3} + t^2_w + W^2_{end} \end{aligned} \tag{3}$$

In the formula (3), the value of expression $2t^1_{send,1}$ is equal to sum $t^1_{send,1} + t^1_{send,3}$, because the size of **Δw** in the case of training set parallelism is the same. Afterwards the epoch price for the number of processing element equal to *n* (t_{iter_n}) is defined as

$$\begin{aligned} t_{epoch\_n} &= \sum_{k=2}^{n}\left[W^1\mathrm{P}_{rec,k} + t^1_{rec,k}\right] + W^1_L\mathrm{P}_{\Delta w,2} + \\ &+ t^1_{\Delta w,2} + t^1_w + W^1_{end} \end{aligned} \tag{4}$$

The formula in (4) is explicitly defining only the time cost of realization of certain computational processes and inter-processor communications. Other elements which define the whole processing time are implicitly occurring in the form of latencies *W*.

3. DESIGN OF SIMULATION MODEL

The simulation architecture consists of the network of workstations (NOW) which is configured by the PAHRA model. There are 24 workstations interconnected by using the 100Mbit Ethernet. The communication between them is provided by the LAM/MPI interface. The synchronizing ring topology is established on the software level by using the C++ language and the LAM/MPI interface.

In the nodes of NOW there were placed workstations with next configurations: A:{P4 2,8GHz:512 MB RAM}, B:{P4 2,4GHz:256MB RAM}. The processing time (t_{epoch_n}) was used as the parameter of performance of simulation architecture. Multilayered feed-forward neural network used in simulation experiments is configured as a three-layer neural network with configuration 256×126×256 (64,512 weight connections), in the role of auto-associated memory. The input synaptic layer (synaptic connections between input and hidden layer) is ensuring coding (compression) of input data of the neural network and the output synaptic layer (synaptic connections between hidden and output layer) is ensuring decoding (decompression) of data. As the training set a grayscale picture (with the dimension 80×32) was used. The training set was divided into 10 sectors of the dimension 16×16 (totally 256 points) which were representing the input and the output vector.

Three experiments were realized. Their description is as follows:

Exp-1. In this experiment a network of homogenous workstations of type A was used (Fig. 4a).

Exp-2. In this experiment workstations of type B were connected to the synchronizing ring topology between workstations of type A (Fig. 4b).

Exp-3. In this experiment workstations of type A and B were cross-connected to the synchronizing ring topology (Fig. 4c).

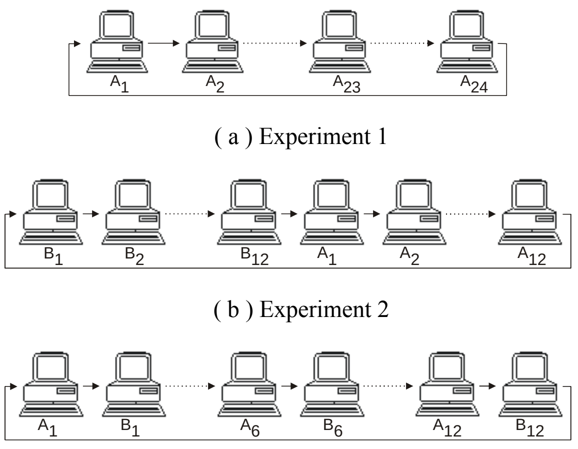
In all experiments was used a mapping scheme based on the proportional distribution of patterns. The number of patterns allocated on PE's was depending on the speed of floating point multiplication operation on given workstation (ALLOC-PR).

3.1. Results of simulation experiments

In the frame of every experiment, two simulations were performed. The goal of the first experiment was to illustrate time flow of realization of one iterating step of BP algorithm with regard to the number of used workstations in designated simulation architecture. Figure 5 shows a time flow of optimal number of workstations $n = 10, 11, ..., 15$. Considering results, the price of realization of one iteration step is lowest with the number of workstations $n = 12$. The second simulation provides information about the rate of speed up (Fig. 6). The speed up (S) was calculated as

$$S(n) = \frac{t_{epoch\_A}}{t_{epoch\_n}} \tag{5}$$

where t_{epoch_A} is the time taken to realization of one iteration step on one A type workstation.

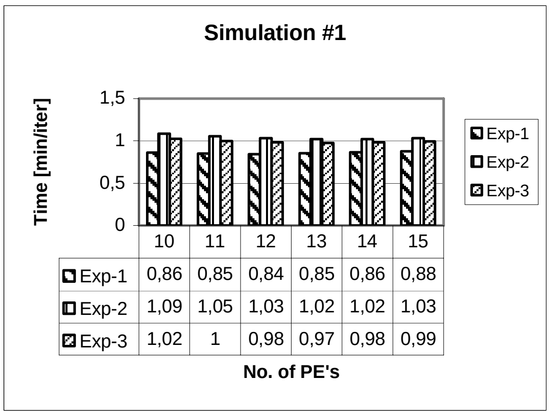


(a) Experiment 1

(b) Experiment 2

(c) Experiment 3

Fig. 4 Simulation architectures for 3 types of experiments



	10	11	12	13	14	15
Exp-1	0,86	0,85	0,84	0,85	0,86	0,88
Exp-2	1,09	1,05	1,03	1,02	1,02	1,03
Exp-3	1,02	1	0,98	0,97	0,98	0,99

Fig. 5 Simulation no. 1

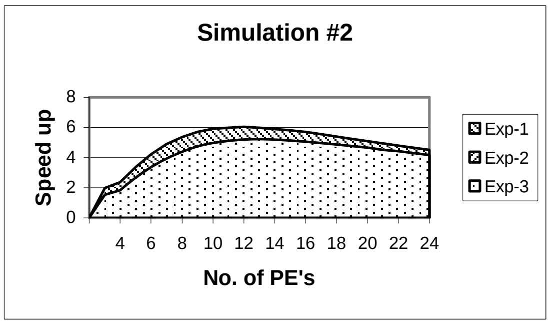


Fig. 6 Simulation no. 2

3.2. Conclusions from simulation experiments

On the basics of results from the simulation experiments, it is possible to write down these conclusions.

- The processing time depends on the network configuration (homogenous; heterogeneous NOW type Exp-2 or Exp-3).
- When the homogenous NOW is used, then the epoch price is lowest.

- In the case of heterogeneous NOW, the configuration depicted on Fig. 4c provides slightly better results than the configuration depicted on Fig. 4b.
- All results show a significant speed up in the learning phase compared to uniprocessor systems.
- The speed up for $1 < n \le 12$ is approximately similar to $n/2$, where n is the number of workstations connected to the parallel simulation architecture based on the PAHRA model.
- The highest speed up for given configuration and solved task, can be achieved with $n = 12$.
- For $n > 12$, the effectiveness is declining.

4. CONCLUSION

The implementation of multilayered feed-forward neural networks represents a very effective way how to reduce the time of learning phase of network. Both phases can be implemented as parallel, but with regard to higher computational load of learning phase, so the research has focused on a parallel approach of this phase. In technical areas there are many application based on artificial neural networks [4]. Most of them are using multilayered feed-forward neural networks with BP algorithm. This combination, together with parallel computer architectures has become a subject of research projects [10] and [11]. Within the frame of these projects a numerical model of parallel architecture PAHRA was designed. This model is bounded with the application of parallel computation in multilayered feed-forward neural network on the level of training set. PAHRA is able to provide a formal description of individual aspects of parallel implementation of multilayered feed-forward neural network and also is able to provide a mathematical tool for verification of performance of selected simulation architecture.

During the simulation experiments, gained results are showing a significant speed up if the learning phase compared with a uniprocessor systems. This speed up reinforces the position of parallel architectures in the modeling process of neural networks, especially in application areas where consequence is focused on the real-time acquisition of required results.

ACKNOWLEDGMENTS

This research has been supported by the Scientific Grant Agency of Slovak Republic under project Vega No.1/1064/04 "*The simulation of parallel computer systems architectures, their specification methods, development technologies and implementations*" and under project Vega No.1/4071/07 "*Intrusion tolerant security architectures of heterogenous distributed and parallel computing systems and dynamic computer networks*".

REFERENCES

[1] Jelšina, M.: Computer system architectures. ELFA s.r.o, Košice, 2002. ISBN 80-89066-40-2. (in Slovak)

[2] Návrat, P., Bieliková, M., Beňušková, Ľ., Kapustík, I., Unger, M.: Artificial Intelligence. Vydavateľstvo STU, Vazovova 5, Bratislava, 2002, ps. 396, ISBN 80-227-1645-6. (in Slovak)

[3] Omondi, A.R. et all.: FPGA Implementations of Neural Networks. Springer, 2006. ps. 360, ISBN-10: 0387284850, ISBN-13: 978-0387284859.

[4] Samarasinghe, S.: Neural Networks for Applied Sciences and Engineering. AUERBACH, 2006. ps. 570, ISBN-10: 084933375X, ISBN-13: 978-0849333750.

[5] Šerbedžija, N.B.: Simulating Artificial Neural Networks on Parallel Architectures. Computer Volume 29, Issue 3, Mar 1996, pages 56-63, ISSN 0018-9162.

[6] Suresh, S., Omkar, S.N., Mani, V.: Parallel Implementation of Back propagatoion Algorithm in Networks of Workstations. IEEE Transactions on parallel and distributed systems, vol. 16. no.1, 2005. pp. 24-34.

[7] Sudhakar, V., Siva, C., Murthy, R.: Efficient Mapping of Back-Propagation Algorithm onto a Network of Workstations. IEEE.Trans. Man, Machine, and Cybernetics—Part B: Cybernetics, vol. 28, no. 6, pp. 841-848, 1998.

[8] Vokorokos, L., Ádám, N., Baláž, A.: Algorithm mapping of MLP network for Neural DF KPI architecture. ICCC 2006, Tallin, Estonia, ISBN 1-4244-0071-6.

[9] Vokorokos, L., Ádám, N., Baláž, A.: Flexible Platform for Neural Network Based on Data Flow Principles. 6th International Symposium of Hungarian Researchers on Computational, Budapest, 2005, Budapest Tech.

[10] Vokorokos, L.: The simulation of parallel computer systems architectures, their specification methods, development technologies and implementations, Project VEGA No. 1/1064/04, DCI FEEI TU of Košice

[11] Vokorokos, L.: Intrusion tolerant security architectures of heterogenous distributed and parallel computing systems and dynamic computer networks, Project No. 1/4071/07, DCI FEEI TU of Košice

BIOGRAPHIES

Liberios Vokorokos, (prof., Ing., PhD.) was born on 17.11.1966 in Greece. In 1991 he graduated (MSc.) with honours at the department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. He defended his PhD. in the field of programming device and systems in 2000; his thesis title was "Diagnosis of compound systems using the

Data Flow applications". He was appointed professor for Computers Science and Informatics in 2005. Since 1995 he is working as an educationist at the Department of Computers and Informatics. His scientific research is focusing on parallel computers of the Data Flow type. In addition to this, he also investigates the questions related to the diagnostics of complex systems. Currently he is dean of the Faculty of Electrical Engineering and Informatics at Technical University. His other professional interests include the membership on the Advisory Committee for Informatization at the faculty and Advisory Board for the Development and Informatization at Technical University of Košice.

Norbert Ádám was born on 30.08.1980. In 1998 he graduated (MSc.) with distinction at the department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. He defended his PhD. in the field of programming device and systems in 2007; his thesis title was "Contribution to simulation of feed-forward neural networks on parallel computer architectures". Since 2006 he is working as a professor assistant on the Department of Computers and Informatics. His scientific research is focusing on parallel computers architectures.

Anton Baláž was born in Sobrance, Slovakia, in 1980. He received the engineering degree in Informatics in 2004 from Faculty of Electrical Engineering and Informatics, Technical University of Košice. Since 2004 he is PhD. student at the Department of computers and informatics FEI TUKE and his scientific research is focused on intrusion detection systems.