Mathematical support and software for data processing in robotic neurocomputer systems

Vitaliy Romanchuk [*]

Ryazan State University named for S.A. Esenin, Department of Informatics, CS and MTI, 390000, Svobody, 46, Ryazan, Russia

Abstract. The paper addresses classification and formal definition of neurocomputer systems for robotic complexes, based on the types of associations among their elements. We suggest analytical expressions for performance evaluation in neural computer information processing, aimed at development of methods, algorithms and software that optimize such systems.

1 Introduction

It is obvious now that productivity of computing devices and systems is insufficient for solving a number of tasks, such as processing of video and audio data, real-time management robotics, recognition of images, forecasting, optimization, and artificial intellect tasks for robotics. The problem is due to several causes: it is impossible to increase the frequencies of computing devices because of the "technological restrictions" we currently face; we lack effective methods, algorithms and software solutions for parallelization of operations when multiprocessor and multinuclear systems are used; we have a limited number of generic methods, algorithms and software means of parallelization, when employing specialized computing devices (neuroprocessors, DSP-processors, etc.).

One of the ways to deal with insufficient productivity in solving a number of tasks is to employ high-performance computing systems with massive parallelism. Such parallelism in numerous applications that will perform complex processing of video, images, communication, security, and other types of signals is mainly present at the data level (Data Level Parallelism – DLP). In the majority of modern processors, DLP is implemented through a single command stream and a multiple data stream, in «single instruction, multiple data» (SIMD). The above applications also show considerable Instruction Level Parallelism (ILP). In such cases, parallel assignment of multiple scalar operations and/or SIMD-operations provides a basis for their parallel performance at the instruction level. However, a SIMD-architecture may not be the best solution for applications with variable DLP, which will decrease productivity and increase power consumption. Modern computing systems use «very long instruction word» (VLIW) architectures that implement ILP and DLP. Nevertheless, popular von Neumann-type processors cannot implement parallelization effectively, because of complex-parallelized constructs in their algorithms, such as cycles, conditional branching, and dependence of data). This paper proposes a method to accelerate the development of methods other than frequency increase, to improve high-speed response: using high-parallel specialized computing hardware (e.g., neurocomputers), and development of innovative easily parallelized algorithms with elements of intellectual compilation. The idea behind this is to simplify the software and extract as much "implicit parallelism" as possible at the command level, using Wide Issue Width (WIW) in command outputs and long pipelines with Deep Pipeline Latency (DPL). Neurocomputers are next generation computing devices consisting of many concurrently running simple computing elements (neurons). These elements are connected in a neuron network. They perform uniform computing operations and do not require external control. The great number of concurrently running computing elements ensures high-response performance and low energy consumption. Besides, neurocomputing applications do not contain any elements that are hard to parallelize, and all computing they do is multiple, parallel, independent, and neuro-based.

The following research teams can be listed as active in similar fields of study:
- S. Pande, F. Morgan, S. Cawley, T. Bruintjes, G. Smit, D. McGinley, S. Carrillo, J. Harkin and L. McDaid [1], D. S. Modha, R. Singh [9], Preisslet [7] who study implementations of certain specific neuro-networks on embedded architectures of neurocomputers, i.e., research of specific architectures.
- N. Izeboudjen, C. Larbes. and A. Farah [2], M. Shulakeret [8], Eliasmithet [4] are active in cataloguing neurocomputing devices, which will further contribute to description of generic properties and principles of functioning for implementation of parallel programming methodology.
- P. Kolinummo, P. PASI Pulkkinen, T. Hämäläinen and J. Saarinen [3], E. Stromatias, F. Galluppi, C. Patterson, S. Furber [5], K. Esseret [6] have chosen a narrow field of research: parallel execution of self-organizing maps (Kohonen maps) on a neurocomputer (emulator).

[*] Corresponding author: v.romanchuk@rsu.edu.ru

2 Mathematical formalization of data processing in robotic neurocomputer systems

Thus, our intention to develop the first universal model of programming for neurocomputing devices and systems.

The goal of the current research is to develop mathematical tools for optimizing parallel, distributed and cloud computing systems based on neuroprocessors.

Our objectives are as follows:
- classification of computing systems built on neuroprocessors, according to associations among their elements;
- formalization of parallel, distributed and cloud computing systems – neuroprocessor computing systems (NPCS);
- defining analytical expressions of performance evaluation in neurocomputer systems of data processing;
- development of optimization methods for parallel, distributed and cloud systems of neurocomputer data processing.
- partitioning of software tools of optimization in parallel, distributed and cloud systems of neurocomputer data processing.

To solve these tasks, we employ the set-theoretic approach, methods of system analysis, methods of optimization and of computing process planning.

We introduce a classification of computing systems based on neuroprocessors, according to types of associations among their elements [4-7].

1. A parallel neuroprocessor computing system is one whose elements are connected on a bus level. The element of a computing system is its computing nucleus, the neuroprocessor. Computing nuclei may refer to specific processors (1 nucleus for each) or to several multinuclear processors. For purposes of simplifying the model, they are regarded as an integrated set of neuroprocessor computing modules (NPCMs) $P=\{P_1,P_2,...,P_q\}$. Processes that are executed on computing nuclei can directly read and write information in the RAM and in the disk storage of the node, through the interface of a system bus or commutator.

A parallel system can be defined in this way:

$$RS=\{P,E,\Delta,S_w,Ttr,Tl\}, \qquad (1)$$

where $P=\{P_1,P_2,...,P_q\}$ is a set of computing nodes of the NCPS, in which each computing node (neuroprocessor) can be described with the following technical specifications:

$$P_i \to H_{ji},H_{ki},H_{li},H_{mi}, \qquad (2)$$

where $H_{ji},H_{ki},H_{li},H_{mi}$ are the technical parameters that define the processor speed, number of its nuclei, RAM and disk storage capacity (if applicable) for the ith NPCM, accordingly;

S_w is the NCPS structure;

E is a set of directional associations between the nodes of the NCPS, whose quantity is defined by the structure type, i.e. $E=F(S_w)$;

Δ - controller (governing node) of the NPCS;

$Ttr: CL\cup\{\Delta\}\times CL\cup\{\Delta\}$ – defines the average throughput between nodes of the NPCS;

$Tl: CL\cup\{\Delta\}\times CL\cup\{\Delta\}$ – defines the latency, i.e., time required to initialize messages, send and receive data, etc.

We can single out the following properties:

$$\forall u,v\in CL\cup\{\Delta\}: Ttr(u,v)=Ttr(v,u)$$

$$\forall u,v\in CL\cup\{\Delta\}: Tl(u,v)=Tl(v,u)$$

As the processors are interconnected through a bus, delay times between data transfer Ttr and latency time Tl are negligible: $Ttr\approx 0$, $Tl=0$.

We can also describe the system's dynamic parameters at any moment of time $t\in[0,+\infty)$:

$U_i(t)$ is the workload on the i th computing nucleus, $0\le U_i(t)\le 1$;

$H_{li}(t)$ is the available RAM of the i th NPCM;

$H_{mi}(t)$ is the available disk storage capacity of the i th NPCM.

An important feature of neuroprocessors is their continuous performance and training (in each command), which consists in data exchange between the RAM and the computing nucleus. This makes it important to correctly choose the external bus configuration for interaction with RAM in a multiprocessor mode. Neuroprocessors, like most DSP processors, support both single-processor and multi-processor work modes along a single or multiple external bus. Let us analyze the most common option, with two buses. If two processors are connected to a shared memory, its access arbitration occurs without an external controller. It should be remembered that only oppositely charged processor buses can unite: a local bus of one processor with the global bus of another, as following a system reset, only one processor will be authorized to access the shared memory.

2. Distributed neuroprocessor computing systems (DNPCSs) are complexes of NPCMs or autonomous computer systems (CSs) that are situated at a distance and interact via programmable commutators and system devices. It should be noted that the rules of parallel and distributed processing are not the same. A GRID system is an example of distributed systems.

One of the features of a distributed system is absence of shared memory, except for a case when a unified address space [5] is provided (we do not discuss it here). Thus, processes that are executed on computing nuclei can directly read and write information in the RAM and disk storage of the node, through the interface of a system bus or commutator.

The notion of cluster is essential in a distributed system. We refer to a neuroprocessor cluster when we mean a group of computers, whose computing nodes are neuroprocessors; the computers have broadband connection and are perceived by the user as a single hardware unit. In other words, a cluster is a loosely connected combination of multiple computing systems that perform jointly in executing shared applications, and are viewed by users as a unified system. A cluster based on neuroprocessors has a number of advantages arising from high-level parallelism of the neurocomputer's paradigm.

Each cluster CL_i can be described in this way:

$$CL_i = \{P_i, K_i, E_i\}, \quad (3)$$

where $P_i = \{P_{i1}, P_{i2}, ..., P_{ig}\}$ is a set of the cluster's computing nodes CL_i;

K_i is a set of commutators;

E_i is a set of associations among them.

The number of computing nodes in set P_i can also equal 1. In cluster CL_i, all computing nodes of a P_i set are connected with a dedicated governing node P_{i0} through a separate governing network that is employed for transfer of signals of government, delivery of program files and input data, and also for output files. Node P_{i0} does not participate in computation. If it is necessary to transfer a message to another computing node within a cluster, a network subsystem is employed; it partitions the message into packages and sends them via a high-speed communication network.

Considering the nature of a neuroprocessor cluster, a distributed system can be described like this:

$$RS = \{CL, E, \Delta, S_w, a, h, q, Ttr, Tl\}, \quad (4)$$

where $CL = \{CL_1, CL_1 ..., CL_r\}$ is a set of computing clusters of the DNPCS;

E is a set of directional associations between clusters of the DNPCS, where some associations can be high-speed (strong), and others slow (loose).

Δ is the controller (governing node) of the NPCS;

S_w is the structure of the DNPCS;

a is the hardware architecture of each DNPCS node;

h is the software architecture of each DNPCS node;

$Ttr: CL \cup \{\Delta\} \times CL \cup \{\Delta\}$ defines the average throughput between each pair of computing clusters (bytes/sec);

$Tl: CL \cup \{\Delta\} \times CL \cup \{\Delta\}$ defines the latency, i.e., time required to initialize messages, send and receive data, etc. between each pair of computing cluster (bytes/sec).

3. Cloud neuroprocessor computing systems (CNPCSs) require ubiquitous access to the network infrastructure of configurable computing resources, such as a neuroprocessor system (we only analyze the *IaaS* service model, where users are given access to the neuroprocessor infrastructure). A significant feature of cloud systems is that there occur time delays due to the employment of relatively slow communication channels between a user and a cloud.

In cloud computing, these two options are possible:

a) The cloud's computing system is not distributed.

b) The cloud's computing system is distributed.

A cloud system can be described like this:

$$RS = \{CL, E, \Delta, S_w, a, h, q, Ttr, Tl, Ttrc, Tlc\}, \quad (5)$$

where $CL = \{CL_1, CL_1 ..., CL_r\}$ is a set of CNPCS computing clusters;

E is a set of directional associations among clusters of the CNPCS, where some associations can be high-speed (strong), and others slow (loose).

Δ is the controller (governing node) of the CPCS;

S_w is the structure of the CNPCS;

a is the hardware architecture of each CNPCS node;

h is the software architecture of each CNPCS node;

$Ttr: CL \cup \{\Delta\} \times CL \cup \{\Delta\}$ defines the average throughput between each pair of computing clusters (bytes/sec);

$Tl: CL \cup \{\Delta\} \times CL \cup \{\Delta\}$ defines the latency, i.e., time required to initialize messages, send and receive data, etc. between each pair of computing clusters (bytes/sec);

$Ttrc$ is the time delay in data transfer to the CNPCS governing device and back to the user;

Tlc is the latency time in data transfer from the CNPCS governing device and back.

If $TO_l^{(j)}$ is the execution time for subprogram $RO_l^{(j)}; l = \overline{1, L}$, defined as the total execution time for type 1 commands,

$$TO_l^{(j)} = TO_{1l}^{(j)} + TO_{2l}^{(j)} \quad (6)$$

then, due to the principles and specific features of neurocomputer functioning, the execution time for auxiliary commands tends to zero: $TO_{2l}^{(j)} \to 0$.

Then $TO_{1l}^{(j)}$ is defined as the total time of mathematical modeling of all neurons in the neuron network of subprogram $RO_l^{(j)}$. If $MK^{(1)}$ is a microcommand for neuron emulation in an artificial neuron network, M_l is the number of commands $MK^{(1)}$ in subprogram $RO_l^{(j)}$.

$$TO_{1l}^{(j)} = M_l * H_k * H_m, \quad (7)$$

where $H_k \in H$ is a technical specification defining the quantity of neurons modeled within a tact;

$H_m \in H$ is a technical specification defining the execution time for one tact of the neuroprocessor.

$$TO_l^{(j)} = M_l * H_k * H_m \quad (8)$$

If Ttr is the total time of data transfer delays between the NPCM in the system; Tl is the total latency time in data transfer.

1. Let us look into the sets that define delay and latency times in data transfer Ttr and Tl for the DNPCS,

$Ttr = \{Ttr_{ij}\}$ where

Ttr_{ij} is the transfer time from i to j NPCS.

The data transfer time can be:

$$\begin{cases} 0 \\ Ttr_{ij} = Ttr_{i0} + Ttr_{j0} \end{cases} \quad (2.72) \qquad (9)$$

where Ttr_{i0} is the data transfer time from node i to the Δ governing node.

$Ttr_{ij} = PS * N$,

where PS is the data throughput for the channel (bit/sec);

N is the bit count for transfer.

$$Ttr = \sum_{i=1}^{|E|} Ttr_{ij}, \qquad (10)$$

$Tl = \{Tl_{ij}\}$, where

Tl_{ij} is the latency time in data transfer between i and j the NPCM.

The latency time can be:

$$\begin{cases} 0 \\ Tl_{ij} = Tl_{i0} + Tl_{0j} \end{cases} \quad (2.72), \qquad (11)$$

where Tl_{i0} is the latency time in data transfer from node i to the Δ governing node;

Tl_{0j} is the latency time in data transfer from the governing node Δ to the j th node.

$$Tl = \sum_{i=1}^{|E|} Tl_{ij}. \qquad (12)$$

$$Ts_{ij} = Ttr_{ij} + Tl_{ij}. \qquad (13)$$

Thus, in distributed systems we obtain a transfer time delay in our NPCS:

$Ts = Ttr + Tl \neq 0$.

In such cases, the initial data transfer to the governing device (commutator) does not cause delays, i.e. $Ttrc \approx 0$.

2. Let us look into the Ttr and Tl sets for the CNPCSs that do not have a distributed structure.

In this case, value $Ttr \approx 0$, and the delay time $Ttrc$ in the system equals:

$$Ttrc = Ttr_{po} + Ttr_{op}, \qquad (14)$$

where Ttr_{po} is the transfer time from the user to the Δ node;

Ttr_{op} is the transfer time from the Δ node to the user.

$$Tlc = Tl_{po} + Tl_{op}, \qquad (15)$$

where Tl_{po} is the latency time in data transfer between the user and Δ;

Tl_{op} is the latency time in data transfer between Δ and the user.

Then we obtain the following:

$$Ts = Ttrc + Tlc \neq 0\,;\; Ttr = Tl = 0.$$

3. Let us look into the Ttr and Tl sets for CNPCSs with distributed structures.

In such cases, the total data transfer times are computed as:

$$Ts = \sum_{i=1}^{|E|} Tl_{ij} + Ttr_{po} + Ttr_{op} + \sum_{i=1}^{|E|} Tl_{ij} + Tl_{po} + Tl_{op} \qquad (16)$$

$$Ts = Ttrc + Tlc + Ttr + Tl \neq 0$$

That is, in this case we obtain $Ttrc \neq 0$; $Ttr \neq 0$.

For further consideration, we assume the total time loss in data transfer to be the following:

$$Tsc = Ttr + Tl + Ttrc + Tlc. \qquad (17)$$

Using the above ratios, we can define the analytical expressions for evaluating the chief criterion of effectiveness: performance of neurocomputer systems for various structures: pipeline, vector, pipeline-vector, and vector-pipeline.

Cycle $T_c^{(j)}$ of the pipeline structure is defined as the duration of the maximum processing time, i.e.

$$T_c^{(j)} = \max_{l \in L} TO_l^{(j)}, \forall l = \overline{1, L}. \qquad (18)$$

The first result of the information processing after implementation of algorithm $A^{(j)}$ will be obtained as the pipeline output after the $t^{(j)} = T_c^{(j)} * q + Tsc$ time.

This value will be the execution time $PR^{(j)}$, i.e.

$$T_o^{(j)} = \max_{l \in L} TO_l^{(j)} * q + Tsc, \forall l = \overline{1, L}. \qquad (19)$$

Each subsequent processing output will be obtained after $T_c^{(j)}$, and then the times of obtaining outputs can be calculated in this way:

$$T_o^{(j)} = \max_{l \in L} TO_l^{(j)} * q + (N-1) * \max_{l \in L} TO_l^{(j)},$$

where N is the ordinal number of the required processing output.

The time loss will be the difference in times between execution time $T_o^{(j)}$ and the total execution time for all subprograms:

$$T_n^{(j)} = q * \max_{l \in L} TO_l^{(j)} - \sum_{l=1}^{q} TO_l^{(j)} \cdot \quad (20)$$

On the other hand, processing each next word will yield a time gain as compared to a single-processor option of the system's implementation $T_l^{(j)} = \sum_{l=1}^{q} TO_l^{(j)}$. Then the gain time will equal the difference between the total operate time of all NPCMs and the total execution time for all subprograms:

$$T_в^{(j)} = \sum_{k=1}^{q} ((k-1) * TO_{q-k+1}^{(j)}) \cdot \quad (21)$$

The downtime can be computed as the sum of differences $T_c^{(j)} - TO_i^{(j)}$ for each i th subprogram in each NPCM:

$$T_{np}^{(j)} = \sum_{k=1}^{q-1} (k * (\max_{l \in L} TO_l^{(j)} - TO_{q-k+1}^{(j)})) \cdot \quad (22)$$

The total processing time is defined as the execution time for all q subprograms on all NPCMs:

$$T_p^{(j)} = \sum_{k=1}^{q} (k * TO_{q-k+1}^{(j)}) \cdot \quad (23)$$

Let us look at the time characteristics for distributed and cloud systems that have a pipeline structure.

For distributed or cloud systems at each pipeline run, data transfer time delays will be added, totaling the delay times of all $q-1$ associations:

$$Ttr = \sum_{i=1}^{q-1} Ts_{i(i+1)} , \quad (24)$$

$$Ts_{ij} = Ts_{i0} + Ts_{(i+1)0} , \quad (25)$$

Then, the DNCPS's pipeline cycle will equal:

$$T_c^{(j)} = \max_{l \in L} TO_l^{(j)} + \max_{Ts} Ts_{ij} \cdot \quad (26)$$

The first result of data processing during the $A^{(j)}$ algorithm implementation will be obtained at the pipeline output after $t^{(j)} = T_c^{(j)} * q + Ts$ time.

Then, considering the delay times, the $PR^{(j)}$ execution time will be:

$$T_o^{(j)} = (\max_{l \in L} TO_l^{(j)} + \max_{Ts} Ts_{ij}) * q \cdot \quad (27)$$

Each subsequent processing result will be the output in $T_c^{(j)}$, so the time when the next processed output is obtained can be computed in this way:

$$T_o^{(j)} = (\max_{l \in L} TO_l^{(j)} + \max_{Ts} Ts_{ij}) * q + + (N-1) * (\max_{l \in L} TO_l^{(j)} + \max_{Ts} Ts_{ij}) . \quad (28)$$

The time lag is defined as the difference in execution time $T_o^{(j)}$ with consideration for time delays in data transfer and the total time of all subprograms' execution:

$$T_n^{(j)} = t^{(j)} - \sum_{l=1}^{q} TO_l^{(j)}, \forall l = \overline{1, L} \cdot \quad (29)$$

The time gain in a distributed or cloud system does not differ from the time gain in a parallel system.

The downtime in a distributed or cloud system:

$$T_{np}^{(j)} = \sum_{k=1}^{q-1} (k * (T_c^{(j)} - TO_{q-k+1}^{(j)})) \cdot \quad (30)$$

The processing time $T_p^{(j)}$ in a distributed or cloud system does not differ from the processing time $T_p^{(j)}$ in a parallel system.

3 Software for data processing in robotic neurocomputer systems

Basing on the obtained analytical expressions, we have developed optimization methods for CSs of neurocomputer data processing that relate to the task of achieving similar processing times for all subprograms and all segments of the program code [7].

For practical study, we chose the *NP Studio* software platform to analyze and optimize the CSs for neurocomputer data processing.

In keeping with the developed classification of structures, we have implemented a choice of five options: vector, pipeline, vector-pipeline, pipeline-vector, and arbitrary architecture.

We have developed a module for modulation, analysis and optimization. The results of this analysis are the following:

1. For each subprogram: the number of commands; number of nul commands; subprogram execution time; processor gain time; processor lag time; processor downtime; processing time; processing time as part of the NPCS; data transfer time; load quotient; average tacts per command; delay percentage; nul command percentage; command left-part nul percentage; delays of decoding, execution, transfer, loading; signals at the input bus, local bus, global bus, weight bus.

2. For NPCS: volume of transferred data; productivity factor, load quotient; system runtime; single-processor runtime; pipeline cycle; NPCS loss time; NPCS gain time; NPCS downtime; NPCS processing time; total NPCS lag time; total NPCS downtime; total NPCS processing time; total lag time; total downtime; total processing time; overall evaluation of NPCS without NPCM; overall evaluation of NPCS.

For work convenience with various source data a task model was implemented, to automatically control electric and mechanical systems in the developed Visual Programming subsystem of the NP Studio software platform. The chief item in the subsystem is the notion of a functional unit, connectable to other functional units to perform certain functions.

The workspace of the software application is divided into functional parts:

1. The area of functional units, instance of which may be transferred to the modeling workspace with the drag-and-drop function. The same area contains control items for visualization and deletion of connections between items. When added to the workspace, each item gets a unique ID, which is a concatenation of the item category, the item number within the category and the index number of the item, e.g., "O2.9" defines an item from category Output Signals, Number, with "9" as its unique ID.

2. The workspace (which, in its turn, can be represented as visual items and connections between them, or as a matrix of connections between items.

The theoretical and practical results of the study have been used for development (in cooperation with the Institute of Machinery Studies, Russian Academy of Science) of a specialized neurocomputing robotic device based on neuroprocessors for automated control of modules in electric-mechanical systems (specifically, hexapods) in a near real-time mode [10-13].

4 Conclusion

The paper proposes theoretical and practical conclusions that represent mathematical tools, as well as algorithms and software that are necessary for optimization of computing systems based on conceptually new computing hardware: neurocomputers. The results obtained can be employed in analysis and optimization of tasks that use multiple neural-based computing, e.g., implementations of neurocomputer systems with automated administration of SEMS modules (Smart electromechanical systems).

The research was carried out within the framework of the assignment for the performance of public works in the sphere of scientific activity within the framework of the initiative scientific project of the state task of the Ministry of Education and Science of the Russian Federation No. 2.9519.2017 / BC on the topic "Technologies for parallel processing of data in neurocomputer devices and systems".

References

1. S. Pande et al., Neural Process Lett., **38**, 131–153 (2013)
2. N. Izeboudjen, C. Larbes, A. Farah, Artif. Intell. Rev., **41**, 491–534 (2014)
3. P. Kolinummo, P. PASI Pulkkinen, T. Hämäläinen, J. Saarinen, Neural Processing Letters, **12**, 171–82 (2000)
4. C. Eliasmith et al., Science, **338**, 1202–1205 (2012)
5. E. Stromatias, F. Galluppi, C. Patterson, S. Furber, The International Joint Conference on Neural Networks (IJCNN), 1–8 (2013).
6. S.K. Esser et al., The International Joint Conference on Neural Networks (IJCNN) (2013)
7. R. Preisslet al., International Conference for High Performance Computing, Networking, Storage and Analysis (SC) (2012)
8. M.M. Shulakeret al., Nature **501**, 526–530 (2013)
9. D.S. Modha, R. Singh, Proc. Natl. Acad. Sci. U.S.A. **107**, 13485–13490 (2010)
10. V. Ruchkin, V. Romanchuk, R. Sulitsa, 2nd mediterranean conference on embedded computing (MECO), 58–61 (2013)
11. V. Ruchkin, V. Fulin, B. Kostrov, V.Romanchuk, E. Ruchkina, 4nd mediterranean conference on embedded computing (MECO), 45–50 (2015)
12. E. Diken et al., Microprocessors and Microsystems, **38,** 947–959 (2014)
13. R. Jordans, L. Jozwiak, H. Corporaal, 3rd mediterranean conference on embedded computing (MECO), 32–35 (2014)