Oleksandr Honcharenko, Heorhii Loutskii

HETEROGENEOUS MULTISPACE DATAFLOW NETWORK

The article discusses a method for constructing a heterogeneous dataflow network based on the concept of a PF-network using the excess de Bruijn topology. The issues of the network structure, the main aspects of functioning, the principle of distribution of tasks and mechanisms for ensuring fault tolerance are considered. A superficial review of load balancing automation using tree decomposition and grain management was also done. A comparison with the original concept was made, the main gains, losses and prospects were identified.

Key words: fault tolerance, excess code, Latin square

Fig.: 7. Tabl.: 1. Bibl.: 6.

Urgency of the research. Currently, high-performance computing is increasingly at a dead end. Increasing the number of nodes increases the nominal performance of the system, as well as power consumption and space requirements, while the real performance barely increases. The reason for this is the problems of parallelism inherent in currently popular control flow systems. The solution to this problem is the transition to another paradigm - dataflow. However, this is not a panacea, because there are a number of dataflow architectures, each of which has its own advantages and disadvantages. To circumvent these problems, it is worth considering not one architecture, but a number of specialized architectural solutions combined into a single heterogeneous system. Of course, to ensure high productivity, such a system must be quite large - this raises the issue of its structure. At the moment, only distributed architecture, clusters and networks are capable of aggregating a large amount of computing resources. Thus, a heterogeneous distributed dataflow system or network is, in fact, the only acceptable solution to the given problem, which makes this issue relevant.

Target setting. The key characteristics for such a network are the following 4: ease of management - allocation of tasks and assignments, ease of searching for free resources, data transfer efficiency and fault tolerance. The issue of grain control is no less relevant. The main target of this research is to ensure these requirements within the framework of the proposed method or to create a basis for their implementation in the future.

Actual scientific researches and issues analysis. At the moment, there are a number of studies devoted to dataflow in the context of high-performance computing.

The application and advantages of dataflow computing in nonuniform networks are considered [1], the concept of an open dataflow network based on Petri net elements is proposed [2], the analysis of algorithms and implementations of systems of this type was performed [3]. At the same time, there is interest in the subject area from companies as well: for example, Maxeler Technologies produces dataflow accelerators based on FPGA, one of the areas of application of which is high-performance computing [4].

Uninvestigated parts of general matters defining. Although the subject area continues to develop, there are still unexplored or ignored points. One of them is the issue of granularity management, which is partially implemented in PF networks. Also, insufficient attention is paid to fault tolerance, which is critical for high-performance systems.

The research objective. The purpose of this study is to improve the efficiency of the PF network in the context of high-performance computing. The tasks of the research are the analysis of PF networks with the de Bruijn structure, an overview of their main properties and application for solving problems and meeting requirements, as well as - a conceptual analysis in comparison with the original solution.

The statement of basic materials. A typical PF network includes 3 management levels that provide dynamic parallelization [2, 5]. All of them are implemented in hardware in the form of corresponding elements and are connected by two highways along which data tokens circulate. This management structure is shown in Fig. 1.

At the upper, file level, the task is presented in the form of an executable file - the so-called p-script of subtasks, which describes complex algorithms that include data distribution (partial processing of arrays, operations on big data). It contains computational tasks of a lower order - p-scripts of formulas, between which there are dependencies on the data. The execution follows a model traditional for dataflow systems: a subtask can be started for execution when the ready condition is fulfilled for it. After execution, the result tokens are returned to the concentrator and activate dependent subtasks.

Similar manipulations occur at the operator level, where parallel operators act as objects. Each such operator is a sequential program - an s-script, processed on a functor - processor of classical architecture.

Fig. 1. Management structure of PF network [5]

Hardware structure of PF network. The hardware components of the highest level in this kind of network are the so-called PF servers, interconnected by any network technology: for example, using Gigabit Ethernet. Each such server is a structurally and functionally complete computer, and consists of PF cells. There are 3 types of cells: A, B and C [2]. Fig. 2 shows the structure of these elements.

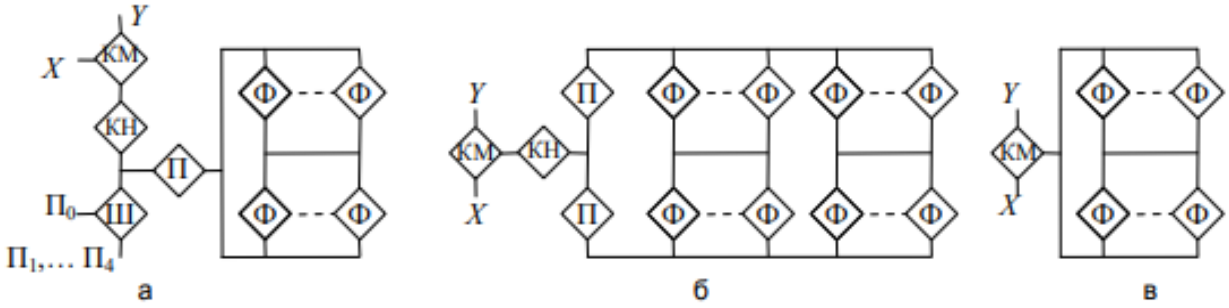

Fig. 2. Structure of PF network elements [2]

The A-cell (foreground cell) is responsible for the distribution of tasks on the system (file level), as well as for the execution of those tasks that require constant uploading of data from external devices. The communicator of this element manages the distribution of tasks and, in case of failure of the background cells, redistributes the files associated with them to other devices.

B-cell (background cell) works mainly at the operator level and performs complex tasks that do not require constant data swapping - for example, processing

arrays. In the event that the B-cell lacks resources, it turns to the processing elements (C-cells), which contain only functors and all processing results are sent to the B-cell.

Combined in a certain way, these elements make up PF servers. There are various options for the structure of their computing field, including two-dimensional and three-dimensional options. In fig. 3 shows examples of the two-dimensional structure of the computing field. On the left is a linear version, where each B-cell corresponds to a line of C-cells. On the right is a matrix, where simultaneous grouping on two axes is performed, and from each B-cell it is possible to reach any processing cell in 2 steps.

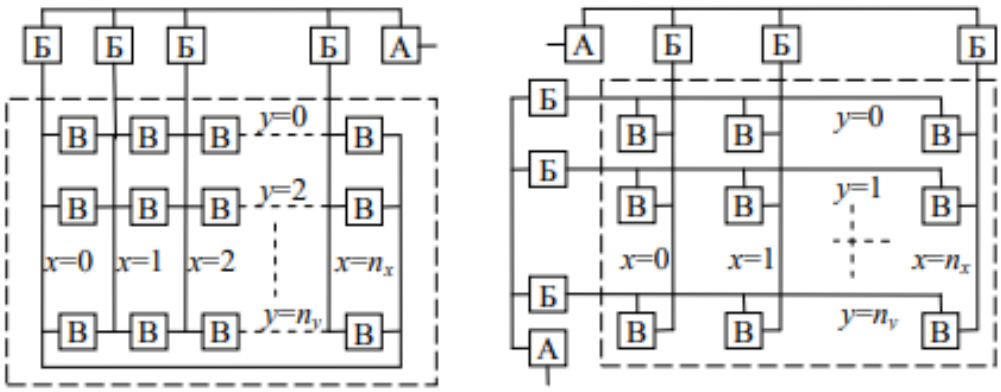

Fig. 3. Examples of the structure of the computing field of the PF server [2].

Advantages of PF network conception. The key advantage of the described solution is its scalability. The PF network is scalable at the level of servers, within each of them it allows additional elements to be connected to the buses, and within each of the elements it realizes the possibilities of separate scaling of the controlling (concentrators, schedulers) and executing (functors) parts. Similarly, this applies to scheduling: the addition of additional resources occurs automatically, and their failure is not critical for calculations and does not require human intervention.

Another interesting aspect is the distribution of management on several levels. Using this kind of dataflow-of-dataflow allows you to fine-tune the grain while sharing overhead between levels.

This makes the proposed solution extremely attractive for the construction of supercomputers and high-performance systems.

Disadvantages of PF network. The key disadvantage of this concept can be called homogeneity and attachment to classic processors as computers. The concept of functors involves the execution of sequential programs according to the MIMD

principle, while ignoring the possibility of combining classical processors with non-classical elements, including graphics processors, vector extensions and specialized FPGA-based chips.

In addition, the extensive use of buses in the structure of servers, on the one hand, provides good opportunities for communication and scaling, but it has its drawbacks. Thus, performing operations in a streaming system requires fairly frequent data transfers between elements, and the bus makes such transfers strictly sequential, unlike a network, where elements are able to exchange information independently of each other.

A final aspect worth noting is the presence of single points of failure in the form of foreground elements. If such an element fails, the entire line of B-cells associated with it, or even the entire PF server, will become unavailable.

De Brujin network as basis. From the point of view of management, the main advantage of this type of network is the simplicity of decomposition into trees. So, the classical de Bruyne topology can be decomposed into 2 binary trees, the excess one into 3 ternary trees [6]. At the same time, for different trees, the sets of non-finite nodes are different, as a result, the root nodes for one tree are finite for others, which allows the use of decomposition both for finding alternative routes and for parallel management of the system. Fig. 4 presents the de Bruyne network, built using the elements of the PF network, as well as the trees into which it is decomposed. At the same time, it is considered that the roots of trees are always A-cells, and other elements of the topology play the role of B-cells. As for C-cells, they are not represented on the topology and are considered abstracted "inside" foreground and background nodes.

In the context of building an effective dataflow network, this kind of property simultaneously solves several problems. On the one hand, independent trees make it possible to load the network with tasks from several directions at once, involving different nodes and in a different sequence, thus avoiding conflicts during transmission. Moreover, the tree-like structure of the hierarchy allows higher-order nodes to abstract from control aspects at lower levels, allowing for more precise control of the load on nodes. On the other hand, network connectivity makes it possible to balance the load on trees by redistributing tasks and subtasks. Another bonus is fault tolerance and static level 4, which makes it possible to implement such a system without unnecessary hardware costs.

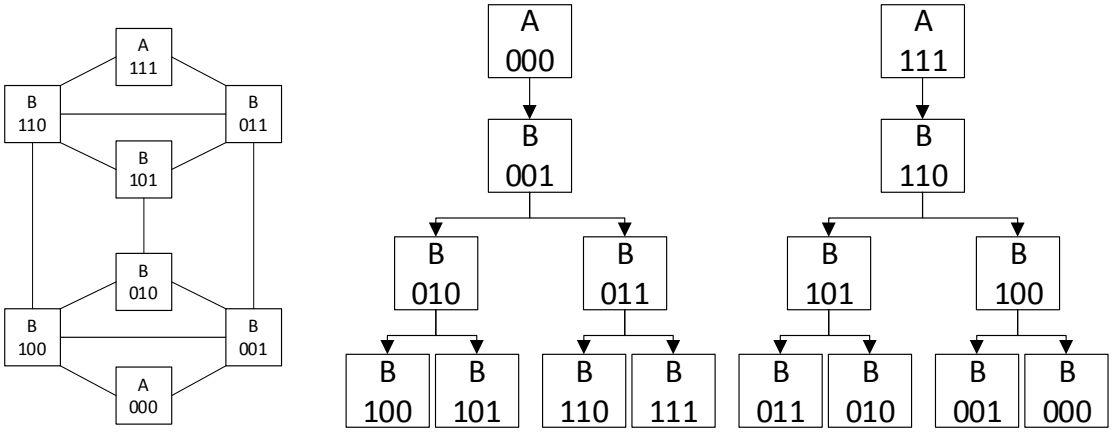

Fig. 4. De Bruyne topology and its decomposition into trees, implemented in the form of a PF network [6]

However, the excess de Brujin network is much more interesting. Unlike the classic one, it uses redundant binary representation to encode nodes (RBR). In fig. 5 shows the model of the PF network based on the redundant topology of de Brujin together with the trees of its decomposition.

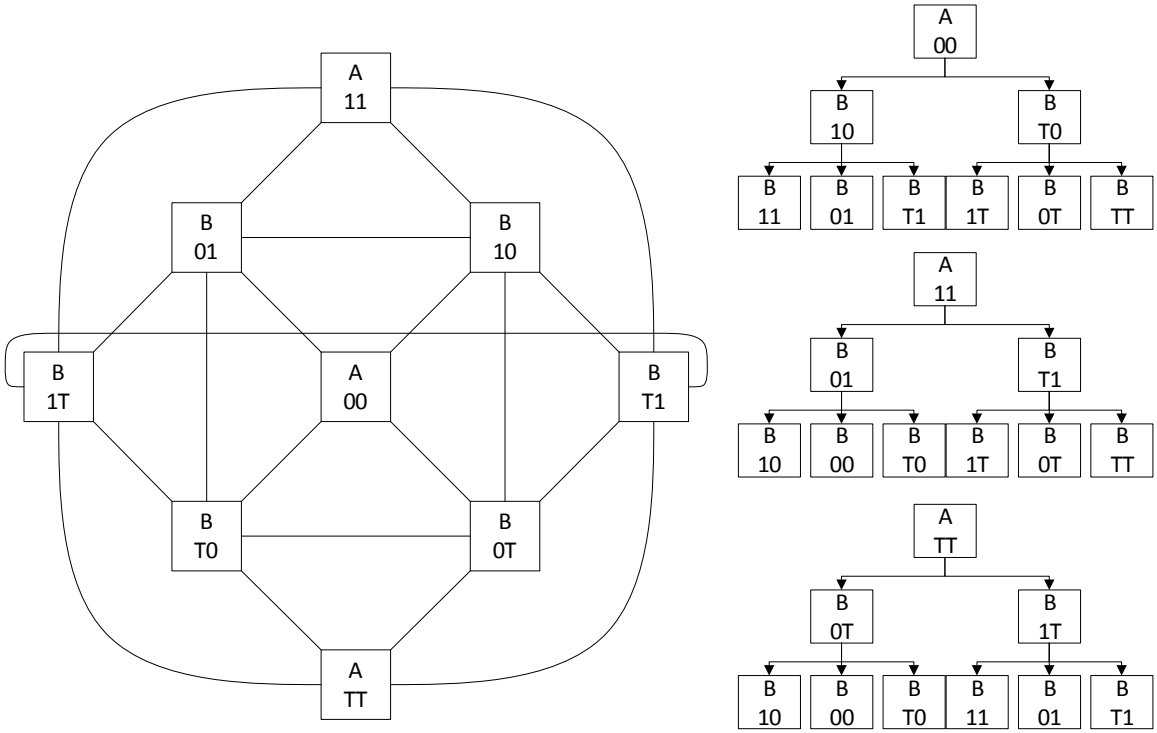

Fig. 5. Excess de Brujin PF network and its decomposition

The specificity of this network compared to the previous one is the property of non-uniqueness of number representations, which can be used to abstract resources

inside a "logical" binary node. Its following properties contribute to this:

1. Each excess de Brujin network contains exactly 2 classical de Brujin subnetworks of the same rank. At the same time, these two subnets always have only one common node - node number 0. For the above network, these are the subnets 00-01-10-11 and 00-0T-T0-TT.

2. In addition to nodes included in 2 subgraphs, there is also a "hidden space" that contains nodes with the same numbers but different codes.

This allows you to logically divide the network into 3 parts - "spaces":

1. An open (direct, positive) space containing nodes that are part of a "positive" binary subgraph. These nodes contain only the numbers 0 and 1 in their code, and are therefore ideal candidates as a "facade".

2. A closed (inverse, negative) space containing the nodes of the "negative" subgraph. Is a complete copy of the "direct" space. This allows you to treat it as a virtual independent device or cluster with the same characteristics as the host system and use it for both balancing and redundancy. At the same time, in contrast to normal redundancy, in this case all working nodes of the main system remain available through redundancy, since they are hardware-only, which makes such redundancy much more effective.

3. Hidden space. Contains nodes with mixed codes. At the same time, due to the properties of RBR, for each hidden node there is one and exactly one node with the same number that is included in the direct or inverse space. This makes it possible to consider hidden nodes as an analogue of C-cells, which expand the computational capabilities of higher-order nodes, but at the same time, each such cell will have its own unique identifier, the node code, which allows it to be accessed not only directly from the owner node, but also and in a bypass, using a common network.

The only exception to this separation is node 0, which enters both open and closed space at the same time. On the one hand, this makes it a good candidate as a main node, but it also makes it the most vulnerable part of the system. This problem can be easily solved using classic redundancy. However, even it is not the only point of failure: thanks to the properties of decomposition, all serviceable nodes of the system are guaranteed to be available, provided that the number of failures in the system does not exceed 2.

Fault tolerance and load balancing. Applying such a separation allows you to apply the following work model. In normal mode, the two parts of the system work

independently. At the same time, node 0 is hardware duplicated: one element serves the "positive" subsystem, and the other serves the "negative". Thus, the two parts of the system will operate relatively independently.

What is the benefit of this? First of all, it allows you to load the system in parallel: on the one hand in 2 spaces at the same time, on the other - in each space from 2 directions. This makes it possible to talk about 4 independent "waves" or "streams" of computing tasks, spreading through the system and filling it with data, starting from the roots of the tree. In fig. 6 shows an example of task distribution over the network.

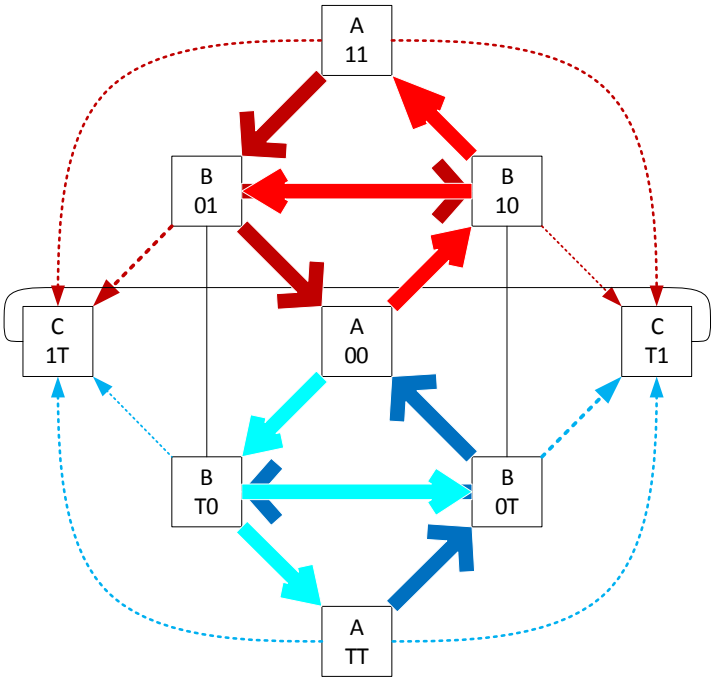

Fig. 6. Distribution of tasks throughout the system.

At the same time, nodes 00, 11 and TT perform the role of A-cells. Other nodes - 01, 10, 0T and T0 - perform 2 roles at once: from the point of view of planning, they are responsible for grinding the grain of the task for transferring tasks to the next level, and from the point of view of calculations - performing those tasks that do not require a large amount of data swapping. As for "hidden" nodes 1T and T1, their role is to calculate specialized tasks and expand the capabilities of neighboring nodes. At the same time, nodes with the same number have priority in accessing their resource.

If a failure occurs, the following is done. If the failure is insignificant (for example, among hidden nodes or in the middle of the tree so that it does not destroy the tree completely) - it can be simply ignored. This will slightly reduce the speed of the system, but

not significantly: since the load streams intersect, the nodes lost to one will immediately become less loaded and more attractive to the other, which automatically balances the load. In the case when the failure is significant, there are several of them and this blocks the very possibility for the stream to continue its work in regular mode - the solution will be to switch to ternary trees. This will allow you to bypass the problem by using hidden nodes and access those parts of the system that are isolated. Fig. 7 shows these 2 cases of failure: on the left - insignificant failure, on the right - transition to ternary trees.

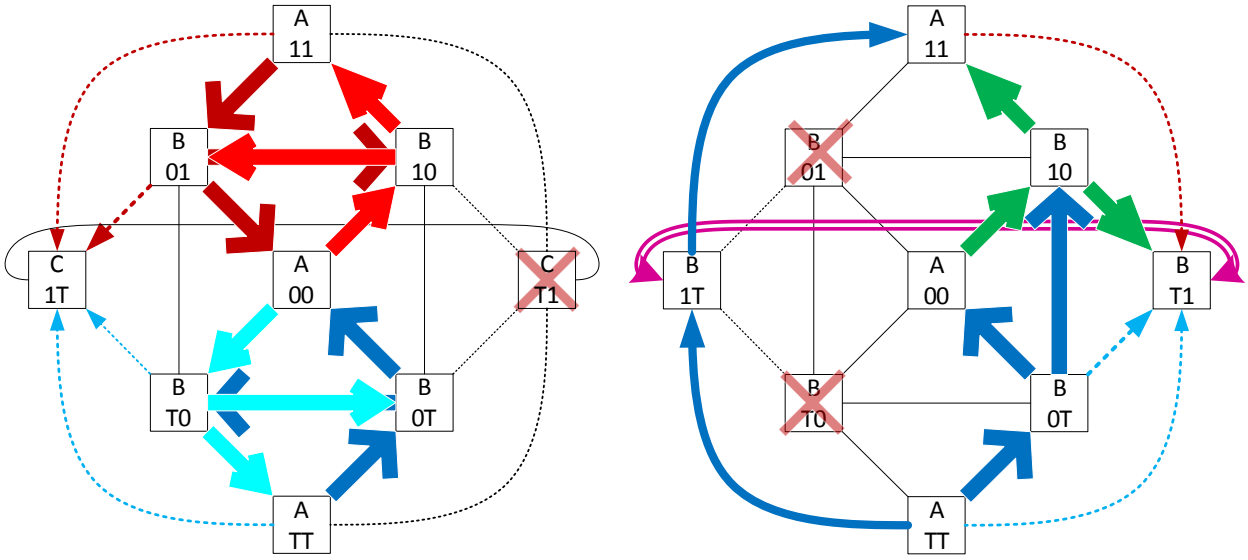

Fig. 7. Solving the problems of fault tolerance.

Results and discussions. Although at this stage of research it is very difficult to evaluate the proposed method, nevertheless, based on the general properties, it is possible to analyze and qualitatively compare the original concept of PF networks and the proposed idea of a multi-space network. Table 1 shows the main characteristics of the system that follow from these two concepts.

Analyzing these properties, it can be seen that the only significant conceptual drawback is the limitation of scaling. This is normal given the topology binding. However, on the other hand, it makes it possible to significantly increase such important characteristics as fault tolerance and ease of resource search, as well as to make shipments much more independent thanks to the combination of bus and direct connections between elements. Another achievement is the distribution of management and loading: a total of 3 nodes manages the system, and as long as at least 1 is functioning, the system is able to fully perform its tasks.

Table 1. **Properties of the system provided by the concept**

Property	Basic SF network	Proposed network
Grain management	On 2 management levels	At 2 management levels and/or at each tree level
Count of loading "streams"	A maximum of 3 with a 3-dimensional field structure	4 in standard mode, 3 – when switching to ternary trees
Roles of cells	Static, defined by the structure of the element.	Partially static: all nodes contain concentrators, schedulers, and functors, but their role is determined by their place in the tree.
Interchangeability of elements	Only within one line	Any element can act as a background.
What is a node in the model?	A specific element – the cell	Abstraction - both a specific cell and a certain group of specialized computers supplemented with control elements of the PF network.
Availability of resources in case of failure	Medium: When A-cells fail, access to associated background cells is lost. However, other cells can be easily replaced if they fail.	High: As long as at least one A-cell is working and connectivity is not broken, all viable nodes will be reachable. Substitution of nodes is also maximal.
Network topology	Multibus	Excess be Brujin with additions
Ease of scaling	Free: elements can be connected to any places of the network without changing its functioning.	Scaling is by topology, so you can't scale the system arbitrarily.
Ease of searching for a free resource	Via bus request	Through tree descent and bus request within the same number
Parallelism of transfers	Low: work on the bus always happens sequentially	High: the only limitation is the impossibility of parallel reception or transmission of 2 messages by one element.

Conclusions. The article proposes a method of constructing a heterogeneous dataflow network, the nodes of which are divided into 3 "spaces" and perform different roles. This allows parallel loading of the system in 4 different directions, treating the 2 halves of the system as relatively independent devices, while allowing devices in one half to access devices in the other.

One of the key advantages of the proposed solution is the good potential for grain control. Since the network can be decomposed into trees (both binary and ternary), this makes it possible to divide the original task into parts - task packages, which will also be divided when descending the tree, which will allow automating load balancing and achieving maximum nodes utilization. Other positive aspects are high fault tolerance and load parallelism, thanks to which the system is filled with tokens not from one, but from 3 points at once, which minimizes latency.

The key disadvantages of the proposed solution are its closedness in scaling, as well as the use of RBR, which complicates the hardware structure.

This approach has great potential for development. A key issue is managing the granularity in task distribution across the tree. Solving this problem will make it possible to implement not only automatic parallelism, but also automatic load balancing and automatic grain management, which may ultimately combine threaded, coarse-grained and classic dataflow, realizing the advantages of each architecture.

References

1. Klimov A. V., Levchenko N. N., Okunev A. S. Benefits of dataflow computation model within nonuniform networks //Informatsionnye Tekhnologii i Vychslitel'nye Sistemy. – 2012. – №. 2. – С. 36-45.

2. Ланцов Р. А. Открытые dataflow-системы с сетевой структурой //Научно-технический вестник информационных технологий, механики и оптики. – 2018. – Т. 18. – №. 6. – С. 1023-1033.

3. Milutinovic V. et al. DataFlow Supercomputing Essentials. – Cham : Springer, 2017.

4. Stroobandt D. et al. An open reconfigurable research platform as stepping stone to exascale high-performance computing //Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017. – IEEE, 2017. – С. 416-421.

5. Ланцов Р. А. Основы параллельного управления в ПФ-сетях //Вестник Казанского государственного технического университета им. АН Туполева. – 2017. – Т. 73. – №. 1. – С. 96-105.

6. Olexandr G. et al. Routing method based on the excess code for fault tolerant clusters with InfiniBand //International Conference on Computer Science, Engineering and Education Applications. – Springer, Cham, 2019. – С. 335-345.

AUTHORS

Honcharenko Oleksandr Oleksiyovytch – PhD student, Department of Computer Engineering**,** National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute” (Solomenskiy district, ave. Pobedy, 37, 03056, Kyiv, Ukraine).

E-mail: alexandr.ik97@ukr.net

ORCID ID: https://orcid.org/0000-0002-9086-6988

Loutskii Heorhii Mychailovytch – professor, PhD, Department of Computer Engineering**,** National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute” (Solomenskiy district, ave. Pobedy, 37, 03056, Kyiv, Ukraine).

E-mail: georgijluckij80@gmail.com

ORCID ID: https://orcid.org/0000-0002-3155-8301