Date of publication JUN-28, 2021, date of current version APR-25, 2021.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.20.2.2168

Effective Distribution of Tasks in Multiprocessor and Multi-Computers Distributed Homogeneous Systems

SERHII ZYBIN[1], VLADIMIR KHOROSHKO[2], VOLODYMYR MAKSYMOVYCH[3], IVAN OPIRSKYY[3]

[1]Department of Computerized Systems Of Information Security, National Aviation University,
1, Liubomyra Huzara ave. Kyiv, 03058, Ukraine (e-mail: zysv@ukr.net)
[2]Department of Information Technology Security, National Aviation University,
1, Liubomyra Huzara ave. Kyiv, 03058, Ukraine (e-mail: professor_va@ukr.net)
[3]Department of Information Technology Security, Lviv Polytechnic National University,
12, S. Bandery str., Lviv, 79000, Ukraine (e-mail: volodymyr.maksymovych@gmail.com, ivan.r.opirskyi@lpnu.ua)

Corresponding author: Serhii Zybin (e-mail: zysv@ukr.net).

ABSTRACT Nowadays, a promising is the direction associated with the use of a large number of processors to solve the resource-intensive tasks. The enormous potential of multiprocessor and multicomputer systems can be fully revealed only when we apply effective methods for organizing the distribution of tasks between processors or computers. However, the problem of efficient distribution of tasks between processors and computers in similar computing systems remains relevant. Two key factors are critical and have an impact on system performance. This is load uniformity and interprocessor or intercomputer interactions. These conflicting factors must be taken into account simultaneously in the distribution of tasks in multiprocessor computing systems. A uniform loading plays a key role in achieving high parallel efficiency, especially in systems with a large number of processors or computers. Efficiency means not only the ability to obtain the result of computations in a finite number of iterations with the necessary accuracy, but also to obtain the result in the shortest possible time. The number of tasks intended for execution on each processor or each computer should be determined so that the execution time is minimal. This study offers a technique that takes into account the workload of computers and intercomputer interactions, and allows one to minimize the execution time of tasks. The technique proposed by the authors allows the comparison of different architectures of computers and computing modules. In this case, a parameter is used that characterizes the behavior of various models with a fixed number of computers, as well as a parameter that is necessary to compare the effectiveness of each computer architecture or computing module when a different number of computers are used. The number of computers can be variable at a fixed workload. The mathematical implementation of this method is based on the problem solution of the mathematical optimization or feasibility.

KEYWORDS optimization; distribution; performance; computing module; multiprocessor; multicomputer; neural network.

I. INTRODUCTION

MULTIPROCESSOR and multi-computer computing systems are a powerful tool for solving problems of large dimension [1, 2]. However, their application raises the problem of efficient distribution of tasks between processors and computers, the essence of which is that two conflicting factors affect the system performance with the specified distribution: load uniformity and interprocessor or intercomputer interactions. Uniform loading plays a key role in achieving high parallel efficiency, especially in systems with a large number of processors or computers. The number of tasks that are intended for each processor or computer

should be determined so that the execution time is minimal. Task's execution time for parallel distribution is defined as the maximum total time that processors or computers need to complete work. It follows that it is necessary to increase the number of processors or computers to minimize the execution time of tasks. However, as Amdahl's law [3] shows, system performance ceases to grow in proportion to the increase in the number of processors or computers used. It turns out that this effect is caused by saturation of interprocessor or intercomputer interactions. Therefore, we must simultaneously take these conflicting factors into account when performing a job allocation in multiprocessor computing systems.

It should be noted that at the present stage of computer technology development, the construction of computer modules based on processors is widely used. They are subsequently combined into multiprocessor computing systems [4–7].

The weak point of almost any computing system is the speed of interaction with memory, because processors during the job request data in memory. It can be fast memory (cache) or Random Access Memory (RAM). The access to RAM is carried out through the data bus due to which there is a decrease in the performance of the computing system. The main feature of the architecture modules is the organization of memory in it both shared and local memory of each processor. Access to shared memory is via a shared bus, through which interaction between processors also takes place. Processors are a shared resource.

Computing modules are a hardware and software unit of the system that implements the main functions (for which the module was created), as well as additional ones (in the process of problem orientation). It provides a change in connections with other modules [6] during reconfiguration of a computer system and the mode of its functioning.

II. RELATED WORKS

Studies [8–10] in the field of efficient use of distributed systems have been ongoing for a long time. They are mainly aimed at solving resource management problems or are focused on the problems of choosing [11] the architecture of a multiprocessor system, or go towards the development of parallel algorithms, resource monitoring and load balancing. This is due to the fact that the efficiency of using distributed systems, in this case, is determined by the possibility of organizing parallel processing [12-14].

Efficiency of using a distributed network is provided due to the maximum load of all resources in order to increase the volume of transmitted traffic. We need to use a rational choice of paths for traffic passing through the network in order to achieve a balanced load of all network resources [15, 16].

Studies are known [17–22], which are devoted to solving the increasing productivity problem. However, the problem in these studies is solved under certain assumptions. Interprocessor and intercomputer interactions are not taken into account when these assumptions are used. The efficiency of the system is achieved due to the uniform loading of processors and computers. In this case, the efficiency of the system is achieved due to the even load of processors and machines: either an equal number of tasks are distributed to all processors and machines in advance, or the system is considered homogeneous. Thus, the efficiency of the system is achieved by minimizing interprocessor, intercomputer interactions.

Other studies [23-25] treated solution through the use of unused computing capacity which resources are unlimited in the volume memory and inexpensive. The method of adaptive load balancing [25] of load balancing is used taking into account the load factor of computers to minimize the time of solving the problem and overhead.

A big problem in research and analysis of multiprocessor systems is that research in this area is mostly classified. For example, we can analyze some information about the systems Elbrus-1, Elbrus-2 [29], however, there are no scientific details in open sources about the principles of optimal distribution of tasks in multiprocessor systems such as Elbrus-16S, Elbrus-2S3 or Elbrus- 12C. Hence, we can conclude that work on the optimal distribution of tasks in such systems is relevant and in demand. For example, the optimal distribution of tasks during a Brute Force attack on containers that are protected by AES encryption (for example, when using a satellite communication channel). In the presence of a multimillion database with passwords, it is possible to reduce the selection time by 1.5-2 times, which is essential in the conditions of military or military operations.

Thus, a comprehensive solution to the problem of minimizing the time to complete tasks is required. Such a solution is proposed to be implemented by the method of efficient distribution of tasks in a distributed computing system, taking into account the workload of processors, as well as interprocessor interactions.

III. MAIN GOAL OF THE ARTICLE AND STATEMENT OF THE PROBLEM

The main goal of the study is to increase the efficiency of the system by minimizing the execution time of tasks and computations. To achieve this goal, a technique is proposed for the efficient distribution of tasks in a distributed computing system, taking into account the workload of processors and computers, as well as interprocessor and intercomputer interactions.

Therefore, it is necessary to solve the following tasks:

– choose an approach to solve the problem of efficient distribution of tasks in a distributed computing system, taking into account the workload of processors and machines;

–make a selection criterion of optimization and limitations;

– improve the communication optimization scheme and conduct a comparative analysis of the research results.

IV. MINIMIZING THE TOTAL TIME FOR COMPLETING TASKS TAKING INTO ACCOUNT THE WORKLOAD OF COMPUTERS AND INTERCOMPUTER INTERACTIONS

We define *m* indivisible (atomic) tasks and a distributed system that consists of *n* computers or processors.

We introduce the following notation:

t_{ij} – is the execution time of the *i*-th task on the *j*-th computer (*i=1, 2, ..., m; j=1, 2, ..., n*);

W_{kl} – is the link weight between computers *k* and *l* (*k, l =1, 2, ..., n*). Link weight is a dimensionless quantity that indicates the priority (rank) of connections between machines, processors (PCs) or interprocessor interaction, etc.;

a_{pq} – is the number of links between tasks *p* and *q* (*p,q=1, 2, ..., m*);

v_i – is the amount of memory needed to solve the *i*-th task;

V_j – is the amount of memory of the *j*-th computer or available memory for the *j*-th processor, and $V_j \geq v_i$.

Without loss of generality, we assume that the number of tasks is greater than or equal to the number of computers $m \geq n$. Otherwise, we introduce additional fictitious tasks $m+1, m+2, ..., n$, assume $t_{ij} = 0$ for them at $i \geq m+1$, and $a_{pq} = 0$ at $p \geq m+1$ or $q \geq m+1$.

Moreover, the goal is to assign each of the *m* tasks to one computer in such a way that the total time for completing tasks taking into account the workload of computers and intercomputer interactions is minimal. It is known that each such assignment is the permutation $(S_1, S_2, ..., S_m)$, which is composed of numbers $(1, 2, ..., n)$, $S_j \in (1, 2, ..., n)$, $(j = 1, 2, ..., m)$. Here it is implied that if *m* = *n*, then $S_i \neq S_j$ at $i \neq j$, i.e. each computer is assigned only one task. If $m > n$, then the case is possible $S_i = S_j$ for $i \neq j$, i.e. multiple tasks can be assigned to one computer.

Currently, there are three approaches to solving the problem: graph-theoretical, heuristic, and mathematical programming method [18, 19].

The graph-theoretical method is based on the geometric representation of the objective function and optimal solutions. Graphs are considered as models of communication networks of multiprocessor and multicomputer systems in the construction, analysis and optimization of such systems. A lot of restrictions, in this case, are implemented using edges, the lengths of which describe the restrictions between pairs of objects and can be represented in two forms: connectivity and minimum distance.

The advantages of the graph-theoretical method are its visibility and simplicity of the solution algorithm.

Heuristic methods for solving problems mean special methods for solving problems, which are usually contrasted with formal methods of solving, based on exact mathematical models [20].

There is no consensus on the number of existing heuristic methods. The basic heuristic methods are:

– a method of dividing the problem into subtasks, using which a complex, non-standard problem is divided into several standard, simple and trivial problems with a known solution;

– a method for introducing auxiliary elements, using which auxiliary elements are introduced in order to eliminate the uncertainty of the relationship between known data and unknown variables that should be found;

– a modeling method, using which the original problem is replaced by its model.

The heuristic method for solving problems can be considered a universal method for finding a solution to a problem. The advantage of using heuristic methods is the reduction of the time for solving a problem in comparison with the method of exhaustive search of alternatives [20, 21]. Heuristic methods increase the likelihood of obtaining a solution to a problem, but this solution is not always the optimal solution. Such a solution to a problem is often a satisfactory solution. Also, heuristic methods are able to find solutions in difficult situations. In addition, in terms of time efficiency, they are not inferior to algorithmic approaches.

The basis of mathematical programming is the mathematical apparatus for solving optimization problems, in which the search for extreme values of the objective function is carried out taking into account constraints [26]. The presence of constraints that are imposed on the values of the objective function makes it impossible to use mathematical analysis methods to solve mathematical programming problems.

Mathematical programming is divided into types of problems that are solved on linear, nonlinear and stochastic. Many methods have been developed to solve mathematical programming problems. A feature of solving problems by methods of mathematical programming is their high demand for computing power, because we have to execute large volumes of computations. Accordingly, considerable importance is attached to the efficiency, simplicity and ease of implementation of methods on computer systems.

In this article, the problem is solved by the method of mathematical programming.

It is obvious that any of the tasks of the *i*-th computer S_i is described by the correspondence $i \rightarrow S_i$ (*i* = 1, 2, ..., m). Moreover, for any tasks, there is, firstly, time, which is equal to t_{iS_i}, and secondly, the time of the interlinks between tasks. Suppose that this time when assigning task *i* to computer S_i and task *k* to computer S_k is equal to the product of the number of links a_{ik} between tasks *i* and *k* by the weight of the link $W_{S_i S_k}$ between computers S_i and S_k, i.e.

$a_{ik}W_{S_iS_k}$. Thus, the task is reduced to obtaining the permutation $(S_1, S_2, ..., S_m)$ composed of numbers (1, 2, ..., *n*), which minimizes the total time

$$\sum_{i=1}^{m} t_{iS_i} + \sum_{i=1}^{m}\sum_{k=1}^{m} a_{ik}W_{S_iS_k} \to \min. \quad (1)$$

We introduce the variable χ_{ij}, which is equal to 1 if the *i*-th task is assigned to the *j*-th computer, and which is zero in the opposite case. Then expression (1) can be written in the following form:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}\chi_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} a_{ik}W_{jl}\chi_{ij}\chi_{kl} \to \min. \quad (2)$$

It follows from this expression if the tasks are independent, i.e. $a_{ik} = 0$, then expression (2) – the minimization task for all permutations transforms into the assignment problem. In addition, vice versa, if all the time values are the same, then we will solve the task of minimizing the total time of interaction, i.e. second term of expression (2).

Since each of the *m* tasks is assigned to only one of the computers, the following condition must be fulfilled

$$\sum_{j=1}^{n} \chi_{ij} = 1, i = 1, 2, ..., m, \quad (3)$$

where

$$\chi_{ij} \in \{0,1\} \forall i, j. \quad (4)$$

On the other hand, the following condition must be met, so that the computer memory is not overloaded

$$\sum_{i=1}^{m} \chi_{ij} v_i \le V_j, j = 1, 2, ..., n. \quad (5)$$

Thus, the statement of the problem is reduced to the task of integer programming $(2) \div (5)$.

For further studies of problem (2), we write this expression in expanded form

$$\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}\chi_{ij} + \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{k=1}^{m} a_{ik}W_{jj}\chi_{ij}\chi_{kj} + \\ + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{\substack{l=1 \\ l\neq j}}^{n} a_{ik}W_{jl}\chi_{ij}\chi_{kl} \to \min. \quad (6)$$

This representation of the expression clearly reflects the essence of the problem, namely, the first two terms express the load of computers, and the third term expresses intercomputer interactions.

Assume that computers are equal in performance and tasks are equal in computational volume. Then the following expression is obvious $t_{ij} = t_0 \forall i, j$. In addition, we assume that $W_{kl} = W_0 \forall k, l$.

In this case, uniform workload can be obtained by distributing the same number of tasks to each computer:

$$\begin{cases} \sum_{i=1}^{m} \chi_{ij} = m_j, j = 1, 2, ..., n; \\ \sum_{j=1}^{n} m_j = m; \\ m_1 = m_2 = ... = m_n \approx \left[\frac{m}{n}\right]. \end{cases} \quad (7)$$

Thus, we conclude that the first two terms in the problem (6) can be omitted taking into account conditions (7), i.e.

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{\substack{l=1 \\ l\neq j}}^{n} a_{ik}\chi_{ij}\chi_{kl} \to \min. \quad (8)$$

As a result, we obtain the integer programming problem (3) ÷ (5), (7) and (8), which minimizes intercomputer interactions due to the uniform load of computers. The condition (5) is essential. This problem, if the condition (5) is not taken into account, fully corresponds to the integer programming problem obtained in [27].

In the general, integer programming problems are NP-complete problems. There are algorithms for solving certain types of problems, which are characterized by polynomial time consuming [26]. For many integer programming problems, there are no convincing arguments in favor of the existence of such decision algorithms, yet. Therefore, they relate to NP-complete classes. In practice, often the solution of such problems requires unacceptably much time and computing resources [26]. For example the use of neural networks [28] with feedback allows us to reduce the solution time, for example the Hopfield network. It is clear that neural networks in the general case do not guarantee globality of the optimal solution to problem (2). However, in practice, it is often required to find one or several local minima within a certain time frame. In this case, the use of neural networks is very effective. Based on this consideration, in order to ensure the practicality of the optimization approach to the clustering problem, a neural network implementation of task (2). In order to synthesize a neural network for solving the optimization problem, we synthesize a triple of the form $\{N, W, T\}$, where *N* is the set of neurons in the network, *W* is

the matrix of synaptic connections and T is the vector of external displacements. In the general case, the task of synthesizing a network is to determine all the components of this triple – the type and number of neurons, the structure of the matrix of connections and the value of its elements, the value of external displacements. We assume that the type and the dynamics model of neural-like elements is determined. Therefore, the problem of network synthesis is reduced to determining the structure of the network, the matrix of connections W and vectors of displacements T, satisfying the target use of the sentized network.

An integral quality of such networks is the ability to determine the state with a minimum level of network energy. In solving problems using neural networks with feedback, the main difficulty consists in constructing the energy function of the network.

Before proceeding to the construction of the energy function of the network, we introduce the following notation:

$$b_{ijkl} = \begin{cases} t_{ij} + a_{ii}W_{jj}, & if\ i = k\ and\ \ j = l; \\ a_{ik}W_{jl}, & if\ i \neq k\ or\ \ j \neq l. \end{cases}$$

We represent the task (2) in a more compact form

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} b_{ijkl}\chi_{ij}\chi_{kl} \to \min. \qquad (9)$$

Now we begin to design the energy function of the network.

In accordance with [29], the energy function that is being developed should be built in such a way that it provides both optimization and compliance with constraints. This step in the process of building an optimized network is to design the energy function of the network. Let's construct this function in the form of a sum, where its individual terms are convex functions that take minimum values for the state of the network. These functions satisfy the considered constraints on the state of the network and minimize the objective function.

Based on this, the component that provides optimization (9) is described as follows:

$$G_1 = -\frac{\alpha_1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} b_{ijkl}y_{ij}y_{kl}\,. \qquad (10)$$

The component that ensures compliance with constraints (3) ÷ (5) is described as follows:

$$G_2 = \frac{\alpha_2}{2}\sum_{i=1}^{m}\left(\sum_{j=1}^{n} y_{ij} - 1\right)^2 + \frac{\alpha_3}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij}\left(1 - y_{ij}\right) + \\ + \frac{\alpha_4}{2}\left(\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij} - m\right)^2 + \frac{\alpha_5}{2}\sum_{j=1}^{n} f^2\left(\sum_{i=1}^{m} y_{ij}v_i - V_j\right), \qquad (11)$$

where

$\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ – are positive constants (by analogy with the traveling salesman problem, an analogue of feasible routes);

y_{ij} – is the output signal of the ij-th neuron of the neural network, which corresponds to the variable χ_{ij};

$f(t) = t - |t|$ – is the function that has the property $f^2(t) = 2tf(t)$.

The first term in the expression (11) corresponds to the constraint that each row of the matrix Y contains exactly one unit; the second term corresponds to the binary variables y_{ij}; the third term corresponds to the restriction that the matrix Y contains exactly m units; and finally, the last term corresponds to constraint (5). Thus, we conclude that when conditions (3) ÷ (5) are satisfied, the component G_2 assumes its minimum value equal to zero.

We perform the summation (10) and (11), after some transformations, we obtain the following form of the energy function of the neural network:

$$G_1 + G_2 = -\frac{\alpha_1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} b_{ijkl}y_{ij}y_{kl} + \\ + \frac{\alpha_2}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} \delta_{ik}y_{ij}y_{kl} - \\ - \frac{\alpha_3}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} \delta_{ik}\delta_{jl}y_{ij}y_{kl} + \\ + \frac{\alpha_4}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} y_{ij}y_{kl} - \qquad (12) \\ - \alpha_2\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij} + \frac{\alpha_3}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij} - \\ - m\alpha_4\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij} + \frac{\alpha_2}{2}m + \frac{\alpha_4}{2}m^2 + \\ + \alpha_5\sum_{j=1}^{n}\left(\sum_{i=1}^{m} y_{ij}v_i - V_j\right) f\left(\sum_{i=1}^{m} y_{ij}v_i - V_j\right),$$

where, δ_{ij} – is the Kronecker symbol.

In expression (12), terms that are independent of the state of the neural network y_{ij}, can be excluded.

The canonical form of the energy function corresponding to problem (9) is written in the following form:

$$G_C = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} W_{ijkl} y_{ij} y_{kl} + \sum_{i=1}^{m}\sum_{j=1}^{n} T_{ij} y_{ij}, \quad (13)$$

where

W_{ijkl} – is the synaptic weight between the input of the *ij*-th neuron and the output of the *kl*-th;

T_{ij} – is the threshold of the *ij*-th neuron.

We compare expressions (12) and (13). We equate the coefficients of their linear and quadratic components. We find the parameters of the neural network

$$\begin{cases} W_{ijkl} = \alpha_1 b_{ijkl} - \alpha_2 \delta_{ik} + \alpha_3 \delta_{ik}\delta_{jl} - \alpha_4; \\ T_{ij} = -\alpha_2 + \dfrac{\alpha_3}{2} - m\alpha_4 + \alpha_5, \end{cases}$$

where

i, *k* = 1, 2, …, *m*;

j, *l* = 1, 2, …, *n*.

Now we calculate the parameters of the neural network that solves problems (3), (4), (7) and (8).

We write problem (8) as follows

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} a_{ik}\left(1-\delta_{jl}\right)\chi_{ij}\chi_{kl} \to \min . \quad (14)$$

Then the energy function of the network for the tasks (3), (4), (7) and (14) takes the following form

$$\begin{aligned} G = &-\frac{\beta_1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} a_{ik}\left(1-\delta_{jl}\right) y_{ij} y_{kl} + \\ &+\frac{\beta_2}{2}\sum_{i=1}^{m}\left(\sum_{j=1}^{n} y_{ij} - 1\right)^2 + \\ &+\frac{\beta_3}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij}\left(y_{ij}-1\right) + \frac{\beta_4}{2}\left(\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij} - m\right)^2 + \\ &+\frac{\beta_5}{2}\sum_{j=1}^{n}\left(\sum_{i=1}^{m} y_{ij} - m_j\right)^2. \end{aligned} \quad (15)$$

Let us compare this expression with (13) and find the parameters of neural networks

$$\begin{cases} W_{ijkl} = \beta_1 a_{ik}\left(1-\delta_{jl}\right) - \beta_2\delta_{ik} + \beta_3\delta_{ik}\delta_{jl} - \beta_4 - \beta_5\delta_{jl} \\ T_{ij} = -\beta_2 + \dfrac{\beta_3}{2} - m\beta_4 - m_j\beta_5, \end{cases} \quad (16)$$

where

$\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ – are positive constants (by analogy with the traveling salesman problem, an analogue of feasible routes) ;

i, *k* = 1, 2, …, *m*; *j*, *l* = 1, 2, …, *n*.

V. DEVELOPMENT OF EFFICIENT DISTRIBUTION OF TASKS IN A DISTRIBUTED COMPUTING SYSTEM

Expression (8) shows that intercomputer communication solves the problem of minimizing task execution time and data calculation.

Since the first priority is the question of intercomputer connections, which determine the cost of time for information transmission, the solution to this problem is possible through the use of neural networks [28]. The use of neural networks also confirmed by [26, 27].

As we said in Section 4, as a result, we got an integer programming problem (3) – (5), (7) and (8), which minimizes intercomputer interactions due to the uniform workload of computers. The solution time can be reduced by using neural networks [28] with feedback.

In the case when several tasks can be assigned to one computer, then when modeling a neural network in addition to the settings using expressions (15) – (16), we need to consider separately the case when m> n. In this case, we need to determine the rate of generation of messages that are transmitted between computers (average frequency of information transfer). At m = n, this parameter depends only on the time of formation of the information parcel by the computer.

Based on the research [5-8,10,11], can be formulated the following consequences.

Consequence # 1. In the model, the number of tasks is much larger than the number of computers. The number of tasks for each computer is equal to $\frac{m}{n}$, the number of tasks external to the computer is equal to $m-\frac{m}{n}$. If similarity between tasks is assumed, then the probability that the task *i* will send a message to the address of the task *j* is equal to $\frac{1}{(m-1)}$ for all *j* = *i*. As a result, we get the following expression

$$\lambda_k = t_\phi\left(m-\frac{m}{n}\right)\frac{1}{m-1} = t_\phi\frac{m(n-1)}{n(m-1)}$$

where

λ_k – is the parameter by which conflict situations for collective resources are evaluated – the rate of generation of messages that are transmitted between computers and is determined as the average frequency of information transfer $\sum_{i=1}^{n} \frac{\lambda_k}{i}$, $i = 1, 2, \ldots, n$;

t_ϕ – time of formation of the information message by the computer.

In addition, if we assume that m is much greater than n, then $\lambda_k = t_\phi \frac{n-1}{n}$.

Consequence # 2. The task of integer programming for a certain time can be realized using neural networks. In this case, the number of computers can be variable $n \in [n_1, n_2]$ in order to increase the reliability of the system. The parameters of the neural network that solves problems (3), (4), (7) and (8) are determined from (15). We will consider the performance of a computing system depending on the method of organizing the interactions between the central processors and the memory, as well as the number of processors in this interaction organization scheme. To simplify the application, we introduce the following constraints:

– the duration of the processing and access to shared resources are subject to the exponential law;

– the processor accesses the shared memory without delay as soon as the shared bus and memory are freed;

– if it is not possible to establish a connection, the processor goes into a standby state and remains in it until the desired resource is released;

– the memory and bus are freed immediately after accessing them, and the processor goes into an active state.

Let present a structural diagram of the organization of connections between central processors and memory. The main feature of scheme # 1 (in fig. 1) for organizing communications is the presence of concentrated memory (organization of cache memory coherency with writeback (MESI protocol [31]) and local memory of each processor. The access to shared memory is via a shared bus.

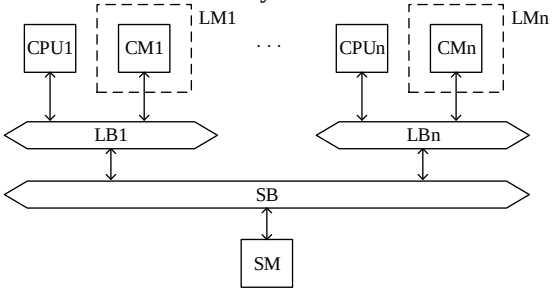

Figure 1. The scheme # 1 for organizing communications: LM – local memory; CM – concentrated memory; SM – shared memory; LB – local bus; SB – shared bus

Shared memory can be divided into local modules for each processor (in fig. 2). It is assumed that local memory is divided into areas of the processor's own memory and shared memory. Moreover, each processor is connected to its own memory via a local bus.

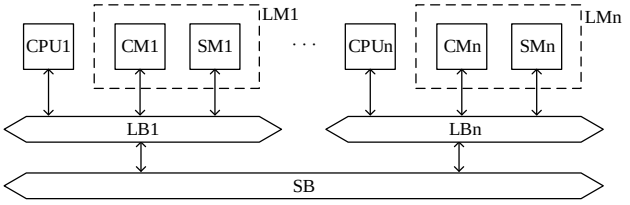

Figure 2. The scheme # 2 for organizing communications: LM – local memory; CM – concentrated memory; SM – shared memory; LB – local bus; SB – shared bus

We will improve the previous scheme. To do this, we will use a multi-port memory module or a multi-level interface for the common part of the local memory of each processor (in fig. 3). Multiport memory eliminates conflicts, but this is achieved by complicating memory. Shared memory modules are directly accessible to external processors via a shared bus.

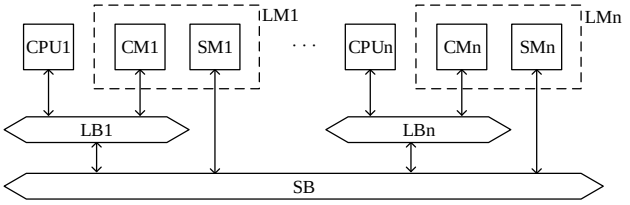

Figure 3. The scheme # 3 for organizing communications: LM – local memory; CM – concentrated memory; SM – shared memory; LB – local bus; SB – shared bus

VI. RESULTS OF THE STUDY, TAKING INTO ACCOUNT THE WORKLOAD OF PROCESSORS AND INTERPROCESSOR INTERACTIONS

When analyzing the schemes presented above, we will use the simulated performance of the computing module (Table 1, Table 2, and Table 3) as a function of the load of processor links for a dual-processor computing module. This is due to the fact that this scheme is a next step for multiprocessor computing systems of a higher rank.

As a result of comparing the scheme # 1 and scheme # 2, we can conclude that, at low loads, the scheme # 1 has higher performance than the scheme # 2. This is because at low loads, the average latency is very small and does not create additional conflicts. In the scheme # 2, each access to the field of external main memory is interrupted by a processor, the probability of activity of which with its own local memory is very high at low loads. The break point is observed at a load value of 0.5. The scheme # 2 becomes more efficient for higher loads.

Table 1. The performance of a dual processor system

The value of processor load connections	The dual processor system performance		
	Scheme # 1	Scheme # 2	Scheme # 3
0.0	1.00	1.00	1.00
0.2	0.65	0.63	0.7
0.4	0.48	0.48	0.54
0.6	0.35	0.36	0.44
0.8	0.29	0.30	0.37
1.0	0.24	0.26	0.31

Table 2. The five-processor system performance

The value of processor load connections	The five-processor system performance		
	Scheme # 1	Scheme # 2	Scheme # 3
0.0	1.00	1.00	1.00
0.2	0.46	0.58	0.65
0.4	0.25	0.39	0.42
0.6	0.16	0.29	0.31
0.8	0.13	0.23	0.24
1.0	0.10	0.19	0.20

Table 3. The eight-processor system performance

The value of processor load connections	The eight-processor system performance		
	Scheme # 1	Scheme # 2	Scheme # 3
0.0	1.00	1.00	1.00
0.2	0.25	0.49	0.51
0.4	0.12	0.25	0.25
0.6	0.09	0.17	0.17
0.8	0.07	0.12	0.12
1.0	0.05	0.10	0.10

Low loads in the computing module, however, can be considered as the most significant. Therefore, a well-designed scheme should function in this area if the goal is to reduce the complexity of distributing tasks between processors by reducing communication costs.

A comparative analysis of the data shows the presence of certain patterns that allow us to draw general conclusions. An increase in the number of processors allows us to ensure that the differences between the schemes become insignificant even for very low loads. For a five- and eight-processor computing module, scheme 3 does not provide such a big advantage over other scheme as in the dual-processor version.

The technique, which is proposed by the authors, makes it possible to compare different architectures of computers and computing modules. In this case, a parameter is used that characterizes the behavior of various models and a parameter that is necessary to compare the effectiveness of each architecture. A parameter that characterizes the behavior of various models is used for a fixed number of computers. The parameter that is necessary to compare the efficiency of each computer architecture or computing module is used when using a different number of computers. The number of computers can be variable at a fixed workload.

Thus, in solving the problems of air traffic control, the application of the proposed method made it possible to reduce the delay time in the processing of incoming information between an airplane and ground ("board-to-ground") by half, which is very important for ensuring flight safety. This methodology is applied at the banks with remote objects. It avoids deadlock situations that arise during operation due to the increase of information flows. Similar situations arise in connection with the transfer of a large amount of information to the central (main) office. Similar situations arise in connection with the transfer of a large amount of information to the central office. Also, the proposed technique has proven itself well in managing complex technological processes that are associated with the processing of statistical information in logistics and making decisions. The mathematical implementation of the method relies on solving the problem of mathematical optimization or feasibility. The practical implementation of the proposed methodology has found application in the field of air traffic control; namely, it is used at the international airport "Borispol", Ukraine, the enterprise "Ukraviarukh" and the state bank "Privatbank".

VII. CONCLUSIONS

This technique allows to optimize task assignments in a distributed system. The technique takes into account the workload of computers and intercomputer interactions (links) and allows one to minimize the time to complete tasks. Parameters λ_k and t_ϕ allow one to compare different architectures of computers and computing modules. At the same time, the parameter λ_k characterizes the behavior of various models with a fixed number of computers. The parameter t_ϕ is necessary to compare the effectiveness of each architecture of a computer or computing module when using a different number of computers, which can be variable, with a fixed workload. The mathematical implementation of the method is based on the solution of the integer programming problem. It is proposed to use neural networks in order to reduce the time for solving tasks.

The application of the optimal task distribution technique allows real-time processing of large amounts of information. At the same time, there is the possibility of automatic parallelization of tasks between computers in the local network. In addition, this technique can be used in systems with remote access.

In the authors' opinion, studies of methods for optimal distribution of tasks require further continuation, because they can be found widely used in processing information in various technological processes.

References

[1] D. A. Patterson, J. L. Hennesy, *Computer Organization and Design: The Hardware/Software Interface: ARM Edition*, Morgan Kaufmann, 2017, 1074 p.

[2] M. Wolf, *Computers as Components: Principles of Embedded Computing System Design*, 3rd Ed., Morgan Kaufmann, Elsevier, 2012, XXIII, 500 p.

[3] F. Gebali, *Algorithms and Parallel Computing*, John Wiley & Sons, 2011, https://doi.org/10.1002/9780470932025.

[4] H. El-Rewini, M. Abd-El-Barr, *Advanced Computer Architecture and Parallel Processing*, John Wiley & Sons, 2005, 287 p. https://doi.org/10.1002/0471478385.

[5] J.-L. Baer, *Microprocessor Architecture from Simple Pipelines to Chip Multiprocessors*, Cambridge University Press, New York, 2010, 384 p. https://doi.org/10.1017/CBO9780511811258.

[6] S.V. Zybin, V.O. Khoroshko, "Productivity and optimization of specialized information processing systems that have a structure, is configured by software," Informatics and Mathematical Methods in Simulation, vol. 9, no. 3, pp. 120–130, 2019, https://doi.org/10.15276/imms.v9.no3.120. (In Ukrainian)

[7] M. Dubois, M. Annavaram, P. Stenström, *Parallel Computer Organization and Design*, Cambridge: Cambridge University Press, 2012, 566 p.

[8] M. Iverson, F. Ozguner, "Dynamic, competitive scheduling of multiple DAGs in a distributed heterogeneous environment," Proceedings of the Seventh IEEE Heterogeneous Computing Workshop, Orlando, Florida USA, March 30, 1998, pp. 70–78.

[9] P. Marshall, K. Keahey, T. Freeman, "Improving utilization of infrastructure clouds," Proceedings of the IEEE / ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'2011), Newport Beach, CA, USA, May 23–26, 2011, pp. 205–214, https://doi.org/10.1109/CCGrid.2011.56.

[10] J. H. Lala, *Performance Evaluation of a Multiprocessor in a Real Time Environment*, Ph.D. Thesis, Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics, 1976. https://dspace.mit.edu/handle/1721.1/61020.

[11] H. El-Rewini, M. Abd-El-Barr, *Advanced Computer Architecture and Parallel Processing*, Wiley-Interscience, 2004, 272 p. https://doi.org/10.1002/0471478385.ch3.

[12] E. De Coninck, T.Verbelen, "Distributed neural networks for internet of things: The big-little approach," Proceedings of the Second International Summit on Internet of Things. IoT Infrastructures: IoT 360°, Rome. Italy. October 27-29, 2015, Revised Selected Papers, Part II, pp. 484-492, https://doi.org/10.1007/978-3-319-47075-7_52.

[13] *Fundamentals of Grid Computing Theory, Algorithms and Technologies, Numerical Analysis and Scientific Computing*, Edited by Frédéric Magoulès, Chapman and Hall/CRC; 1 st edition, 2009, 322 p. https://doi.org/10.1201/9781439803684-c1.

[14] N. Kussul, L. Hluchy, A. Shelestov, S. Skakun, O. Kravchenko, M. Ilin, Yu. Gripich, A. Lavrenyuk, "Data fusion grid segment," *Space Science and Technology*, vol. 15, no. 2, pp. 49–55, 2009, https://doi.org/10.15407/knit2009.02.049.

[15] G. Capannini, F. Silvestri, and R. Baraglia, "K-model: A new computational model for stream processors," Proceedings of the 2010 IEEE 12 th International Conference on High Performance Computing and Communications, HPCC'2010, 2010, pp. 239–246, https://doi.org/10.1109/HPCC.2010.22.

[16] Proceedings of the 15-th International Workshop on Heterogeneous Wireless Networks (HWISE-2019), Kunibiki Messe, Matsue, Japan, March 27–29, 2019. URL: http://voyager.ce.fit.ac.jp/conf/hwise/2019/.

[17] X. Lu, L. Chen, & Z. Li, "Performance evaluation and enhancement of process-based parallel loop execution," *Int J Parallel Prog*, vol. 45, pp. 185–198, 2017, https://doi.org/10.1007/s10766-015-0394-1.

[18] H. Gao, A. Schmidt, A. Gupta, P. Luksch, "Load balancing for spatial –grid-based parallel numeric simulations on clusters of SMPs," *Proceeding of the 11th Euromicro Conference on Parallel, Distributed and Network based Processing (PDP'2003)*, Genoa, Italy February, 5–7, 2003, pp. 75–82.

[19] H. Gao, A. Schmidt, A. Gupta, P. Luksch, "A graph-matching based intra-mode task assignment methodology for SMP clusters," *Proceeding of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI'2003)*, Orlando, Florida, USA, July, 27–30, 2003, pp. 406 – 411.

[20] C. R. Kothari, *Research Methodology: Methods and Techniques*, Second revised edition, New Age International, 2007, 414 p.

[21] C. Andre, R. Pinheiro, F. McNeill, *Heuristics in Analytics: A Practical Perspective of What Influences Our Analytical World*, John Wiley & Sons Inc, 2014, 256 p. https://doi.org/10.1002/9781118434260.

[22] G. Karypis, V. Kumar, "Unstructured tree search on SIMD parallel computers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, issue 10, pp. 1057–1072, 2004, https://doi.org/10.1109/71.313122.

[23] M.A. Miroshnik, L.A. Klimenko, "Placement of subtasks in distributed computing systems of cluster-metacomputing type," *Information and Control Systems on Railway Transport*, no. 4, pp. 71–77, 2014. (in Russian)

[24] H. Shafiee, M. N. Moqadam, "Information resources management (IRM): The key of accountability," Jurnal Fikrah. Jilid 8, Special Issue 1, pp. 232–246, 2017.

[25] A. D. Kshemkalyani, M. Singhal, *Distributed Computing Principles, Algorithms, and Systems*, Cambridge University Press, 2008, 754 p. https://doi.org/10.1017/CBO9780511805318.

[26] W. L. Winston, *Introduction to Mathematical Programming Operations Research: Volume One*, 4th Edition, Thomson Brooks/Cole, 2003, 924 p.

[27] Y.S. Yvanchenko, V.A. Khoroshko, "Analysis of information resource traffic," *Informational security*, no. 1, pp. 63–68, 2013. (in Russian)

[28] L. Li, K. Ota and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4665-4673, 2018, https://doi.org/10.1109/TII.2018.2842821.

[29] V. Dudykevych, I. Prokopyshyn, V. Chekurin, I. Opirskyy, Yu. Lakh, T. Kret, Ye. Ivanchenko, I. Ivanchenko, "A multicriterial analysis of the efficiency of conservative information security systems," *Eastern-European Journal of Enterprise Technologies. Information and Controlling System*, vol. 3, no. 9(99), pp. 6-13, 2019, https://doi.org/10.15587/1729-4061.2019.166349.

[30] L. Karpov, V. Feldman and A. Sheerai, "Universal engineering console and its software for Elbrus-1 and Elbrus-2 multiprocessor computer systems," *Proceedings of the 2017 Fourth International Conference on Computer Technology in Russia and in the Former Soviet Union (SORUCOM)*, Zelenograd, 2017, pp. 54-63, https://doi.org/10.1109/SoRuCom.2017.00014.

[31] J. Gómez-Luna, E. Herruzo, & J. I. Benavides, José, "MESI Cache Coherence Simulator for Teaching Purposes," *CLEI Electron. J.*, vol. 12, no. 1, 2009, https://doi.org/10.19153/cleiej.12.1.5.

SERHII V. ZYBIN *was born in 1970 in Mykolaiv, Ukraine. In 1994, he graduated the Kyiv Polytechnic Institute. The diploma specialty is a system engineer. In 2003-2004 worked in the National Aviation University in the position of assistant. In 2004-2019 worked in the State University of Telecommunications in following positions: senior lecturer, associate professor. From 2019 for this time works in the National Aviation University in the position of professor. In 2006, he received his Ph.D degree. The specialty is "Information security systems". In 2019, he received his doctor degree. The specialty is "Information technologies". Research interests: information security systems, decision support systems, cybersecurity, operating systems.*

VOLODYMYR O. KHOROSHKO *was born in 1945 in Kharkiv, Ukraine. In 1968 he graduated from the Kyiv Institute of Civil Aviation Engineers with a degree in "Technical operation of aviation electronic equipment." In 1975 he defended his Ph.D dissertation, and in 1992 he defended his dissertation for the degree of Doctor of Technical Sciences at the Institute of Modeling Problems in Energy of the Academy of Sciences of Ukraine (Kyiv). He works as a professor at the Department of Information Technology Security of the National Aviation University. Research interests: computer systems and information security: technical systems of information protection, which are complex systems with decision support subsystems; intellectualization of modeling, management and decision-making in the field of information security.*

Volodymyr N. Maksymovych *was born in 1952 in Lviv, Ukraine. In 1975 he graduated from the "Lviv Polytechnic Institute". Specialty after a diploma is electrical engineering. In 1975-1993 worked in the Lviv Research Radiotechnical Institute on positions: engineer, senior engineer, leading engineer, leading engineer-designer from 1992 for this time works in the National University "Lviv Polytechnic", on positions: assistant, associate professor, professor, head of department of Information Technologies Security. In 1993 received his Ph.D. on specialty "Devices and methods of electric and magnetic values measuring". In 2008 received doctor degree on specialty "Elements and devices of the computing engineering and control systems". Research of interests: generators of pseudorandom numbers and bit sequences, number-pulse functional converters, devices for radiation parameters measuring.*

Ivan R. Opirskyy*: In 2008 he graduated from the National University "Lviv Polytechnic" and received a Master degree in "security of information with restricted access and automation of its quitrents." In 2012 he received his PhD. on specialty "Information security systems" at the National University "Lviv Polytechnic". In 2018 received the degree of Doctor of Science in the specialty "Information Security Systems". 2019 is professor of information security department the National University "Lviv Polytechnic".*