Contents

Vol. 38, No. 3, 2012

A simultaneous English language translation of this journal is available from Pleiades Publishing, Ltd. Distributed worldwide by Springer. *Programming and Computer Software* ISSN 0361-7688.

ISSN 0361-7688, Programming and Computer Software, 2012, Vol. 38, No. 3, pp. 143–149. © Pleiades Publishing, Ltd., 2012.
Original Russian Text © N.S. Kovalenko, P.A. Pavlov, 2012, published in Programmirovanie, 2012, Vol. 38, No. 3.

Optimal Grouping Algorithm of Identically Distributed Systems

N. S. Kovalenko [a] **and P. A. Pavlov** [b]

[a] *Belarus State Economic University, Partizanskii pr. 26, Minsk, 220070*

[b] *Polessky State University,@ul. Dneprovskoi flotilii 23, Pinsk, 225710*

E-mail: kovalenkons@rambler.ru, pin2535@tut.by

Received October 1, 2011

Abstract—An algorithm for optimal grouping blocks of a structured program resource in systems of distributed competing processes is proposed.

DOI: 10.1134/S0361768812030036

1. INTRODUCTION

Scalability is one of the most important requirements to modern computational multiprocessor systems (MS), computer systems (CS), databases, routers, etc. It implies that a system is capable of increasing its performance under changes of hardware and program resources. Scalability is now of interest for both designers of parallel multiprocessor systems and a distributed environment of metacomputing [1]. In such computer systems, the use of a shared program resource (PR) [2] is hindered because of the autonomy of processor nodes and the lack of a unified administration policy. The distribution of data and computation processes with the use of structuring and pipelining [2] is a general feature that makes it possible to improve performance of scalable computer systems. In connection with this, there is a need in new computing and resource allocation principles, development of efficient hardware and software, and optimal planning and allocation of computation processes [3]. Efficient control of the set of processes that have access to shared resources (including, program ones) is the task of special importance. A mathematical statement of such problems was proposed and analyzed in [2, 4–6].

In particular, it was shown in [2, 4–6] that the optimal in terms of the number of processes system of distributed competing processes is found among identically distributed systems and uniform structurings of a program resource into parallel running blocks. The uniformity of structuring, however, cannot always be achieved in practice, which makes one seek for alternative approaches. One of them involves constructing optimal groupings from consecutive blocks of distributed processes (all basic concepts and definitions including model parameters are given bellow).

In the paper, an algorithm for the optimal grouping of distributed processes competing for a linearly structured program resource is proposed. The algorithm requires no more than $O(n^3)$ elementary operations, where n is the number of computation processes of an initial identically distributed system.

2. BASIC CONCEPTS AND PROBLEM STATEMENT

As in [2, 4–6], we consider a **process** as a sequence of blocks (commands, procedures) $Q_1, Q_2, ..., Q_s$ that are executed on a set of processors (processor nodes, processing units, intelligent clients). A process is called **distributed** if all the blocks or some of them are processed by different processors. In order to increase performance, processes can be processed in parallel by exchanging information. Such processes are called **cooperative** or **interacting** processes.

The concept of a resource is employed to denote any objects of a computer system that can be used by processes for their own execution. **Reenterable** (reusable) resources can simultaneously be used by several computation processes. In the case of parallel systems, it is often required to repeatedly execute the same block sequence or its part. We will call such a sequence a **program resource** and the corresponding processes **competing** processes.

A mathematical model of a system of distributed processing of competing processes involves the following: s, $s \geq 2$ is the number of blocks of a linearly structured program resource $PR = (Q_1, Q_2, ..., Q_s)$; n, $n \geq 2$ is the number of competing processes distributed with respect to PR; p, $p \geq 2$ is the number of processors in a multiprocessor system; $T_p = [t_{ij}]$ is the matrix of times of executing jth blocks by ith competing processes $i = \overline{1, n}$, $j = \overline{1, s}$; and ε is the time characterizing extra

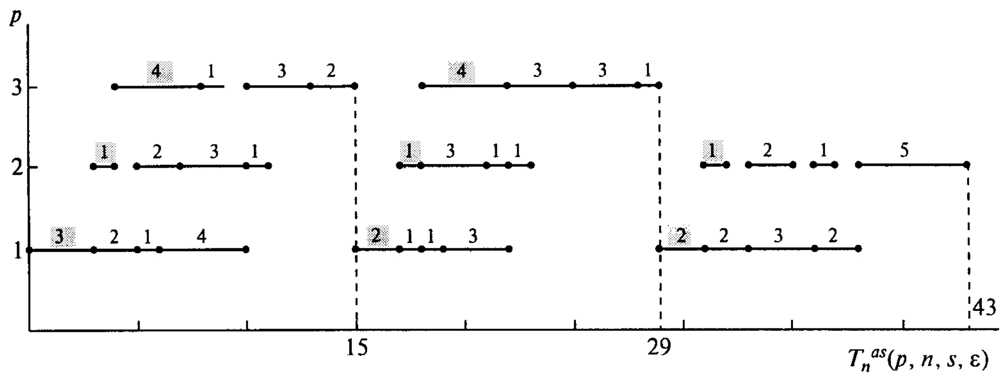

Fig. 1. Asynchronous mode, the unsuperposed Gantt diagram.

system costs on structuring and parallel use of blocks of the program resource *PR*.

As in [4–6], we assume that the interaction between the processes, processors, and blocks of the linearly structured program resource is subjected to the following conditions: (1) none of the blocks *PR* can be processed by more than one processor at a time; (2) none of the processors can process more than one block at a time; (3) each block is processed without interruptions; and, (4) for each process, the blocks of the program resource are distributed over the MS processors cyclically according to the following rule: the block with the number $j = kp + i, j = \overline{1, s}, i = \overline{1, p}, k \geq 0$ is distributed to the processor with the number i. In addition, we introduce extra conditions specifying modes of interaction between the processes, processors, and blocks of *PR*: (5) there are no downtimes of processors provided that the blocks are ready, and there are no non-executions of blocks if processors are available; (6) for each of n processes, the time of completing the jth block on the ith processor coincides with the time of beginning the execution of the next $(j + 1)$th block on the $(i + 1)$th processor, $i = \overline{1, p-1}$, $j = \overline{1, s-1}$; and (7) for each block of the structured program resource, the time of its completion by the lth process coincides with the beginning of its execution by the $(l + 1)$th process on the same processor, $l = \overline{1, n-1}$.

Conditions (1)–(5) describe **an asynchronous mode** of interaction between processes, processors, and blocks, which assumes that there are no downtimes of the MS processors provided that the blocks are ready, and there are no non-executions of blocks if processors are available.

Conditions (1)–(4) and condition (6) constitute **the first synchronous mode** that provides continuous execution of program resource blocks within each computation process.

The second synchronous mode described by conditions (1)–(4) and (7) provides continuous execution of each block by all the processes.

Definition. A distributed system of n cooperative competing processes is called **heterogeneous** if execution times of the *PR* blocks depend on the amount of data to be processed and/or their structure, i.e., are different for different processes.

Definition. A system of cooperative competing processes is called **identically distributed** if times t_{ij} of executing blocks $Q_j, j = \overline{1, s}$, of the program resource *PR* by each ith process are identical and equal to t_i for all $i = \overline{1, n}$, i.e., the chain of equalities $t_{i1} = t_{i2} = \ldots = t_{is} = t_i$ holds for all $i = \overline{1, n}$.

Below is an example of a system of distributed processing of competing processes for the asynchronous mode of interaction between processes, processors, and blocks and its mapping by means of linear Gantt diagrams. Figures 1 and 2 show non-superposed and superposed linear diagrams reflecting the execution of $n = 4$ heterogeneous distributed competing processes on the multiprocessor system with $p = 3$ processors for $s = 8$ blocks of a structured program resource. Execution times for each block are shown in the diagrams and given by *(3,1,4,2,1,4,2,1)*, *(2,2,1,1,3,3,2,2)*, *(1,3,3,1,1,3,3,1)*, and *(4,1,2,3,1,1,2,5)* for the first, second, third, and fourth processes, respectively. For clearness, the times of the first process' blocks are highlighted. The overall execution time $T_{@}^{ac}(p, n, s, \varepsilon)$ of heterogeneous distributed competing processes is equal to *43* and *35* in the cases of the non-superposed and superposed diagrams, respectively.

Let $T_\varepsilon^n = \sum_{i=1}^{n} t_i^\varepsilon$ be the overall time of executing each block Q_j by all n processes with regard to the overheads ε, $t_{\max} = \max\limits_{1 \leq i \leq n} t_i^\varepsilon$, $t_i^\varepsilon = t_i + \varepsilon$, $i = \overline{1, n}$.

It was proved in [6] that, for identically distributed systems of competing processes, the minimum overall

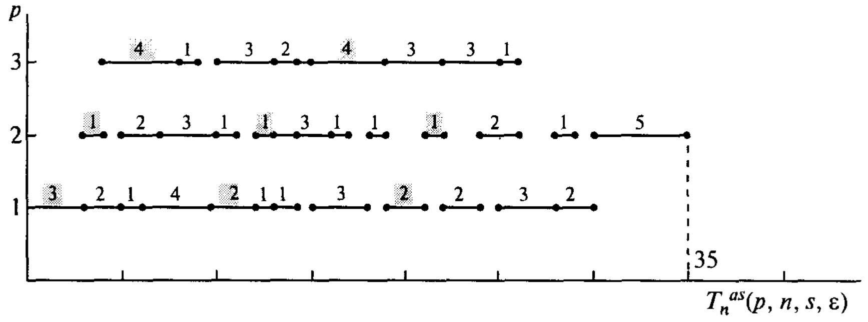

Fig. 2. Asynchronous mode, the superposed Gantt diagram.

time for all the three base modes in the case of **unlimited parallelism** ($s \le p$) and in the case of **limited parallelism** ($s > p$) provided that $T_\varepsilon^n \le pt_{\max}^\varepsilon$ is determined by the following formula:

$$T(p, n, s, \varepsilon) = T_\varepsilon^n + (s-1)t_{\max}^\varepsilon, \qquad (1)$$

while in the other cases, the overall execution time of n identically distributed processes that use the program resource PR structured into s blocks in the computing environment with p processors for the asynchronous mode and the mode of continuous execution of each block by all the processors is given by

$$T(p, n, s, \varepsilon)$$
$$= \begin{cases} kT_\varepsilon^n + (p-1)t_{\max}^\varepsilon, \\ \text{for } s = kp, \quad k > 1, \quad T_\varepsilon^n > pt_{\max}^\varepsilon, \\ (k+1)T_\varepsilon^n + (r-1)t_{\max}^\varepsilon, \\ \text{for } s = kp + r, \quad k \ge 1, \quad 1 \le r < p, \quad T_\varepsilon^n > p_{\max}^\varepsilon, \end{cases} \qquad (2)$$

where $T_\varepsilon^n = \sum_{i=1}^n t_i^\varepsilon$ is the overall execution time of each block Q_j by all n processes with regard to overheads ε, $t_{\max}^\varepsilon = \max\limits_{1 \le i \le n} t_i^\varepsilon$, $t_i^\varepsilon = t_i + \varepsilon$, $i = \overline{1, n}$.

We relate the concept of a set of identically distributed competing processes and the concept of a linear stowage.

Definition. Let $M = \{m_1, m_2, \ldots, m_n\}$ be a finite ordered set of items. A partition of the set M into l disjoint subsets $M_1, M_2, \ldots, M_l$ such that each subset is an assembly of serial elements of the set M is called a **linear stowage** of the set M of rank l.

By the assumption, times of executing program resource blocks by each process coincide $t_{i1} = t_{i2} = \ldots = t_{is} = t_i$, $i = \overline{1, n}$. Therefore, we consider the sequence of the first blocks $(Q_{11}, Q_{21}, \ldots, Q_{n1})$ as elements of the set M and denote this sequence as $(q_1, q_2, \ldots, q_n)$. In this case, the linear stowage of the set M is obtained by integrating blocks q_i, $i = \overline{1, n}$ of the serial processes into a single program block. A linear stowage of blocks q_i, $i = \overline{1, n}$, that results in reduction of the number of processes in the multiprocessor system is called a **linear grouping** and denoted by LC_l.

Let us denote the set of various groupings of blocks in the system of identically distributed processes competing for the use of the program resource PR by K, and the set of groupings of rank l, $l = \overline{1, n}$ by K_l. Note that the grouping of rank n is the initial identically distributed system $LC_n = (q_1, q_2, \ldots, q_n)$, while the grouping of rank 1 is the grouping of blocks into a single program block $LC_1 = (q_1 \cup q_2 \cup \ldots \cup q_n)$. It is easy to calculate that $|K| = 2^{n-1}$ and $|K_l| = C_{n-1}^{l-1} = \frac{(n-1)!}{(l-1)!(n-l)!}$.

Let $LC_l = (q_1', q_2', \ldots, q_l')$ be a linear grouping of blocks.

We introduce the following notation:

$t(q_i') = \sum_{q \in q_i'} t(q)$ is the execution time of the ith element of the grouping LC_l, $i = \overline{1, l}$;

$t(LC_l) = (t(q_1'), t(q_2'), \ldots, t(q_l'))$ is a sequence of the execution times of blocks q_i', $i = \overline{1, l}$;

$t_{\max}(LC_l) = \max\limits_{1 \le i \le l}\{t(q_i')\}$ is the execution time of the greatest block of the grouping LC_l; and

$t_{\min} = \min\{t_{\max}(LC_l) | LC_l \in K_l\}$.

The problem of optimal grouping of blocks $(q_1, q_2, \ldots, q_n)$ of a set of identically distributed competing processes consists in finding a linear grouping LC_l of an initial identically distributed system for given $p \ge 2$, $n \ge 2$, $s \ge 2$, and $\varepsilon > 0$ on which the minima of functionals (1) and (2) are attained. Such a grouping is said to be **optimal.**

3. FEATURES OF OPTIMAL GROUPINGS AND AUXILIARY RESULTS

To solve the posed problem, we need the following results.

Theorem 1. If LC_l is the optimal linear grouping of an identically distributed system, then the grouping LC'_l such that $t_{\max}(LC'_l) = t_{\min}$ is optimal as well.

Proof. It follows from the definition of $t_{\min}$ that

$$t_{\max}(LC_l) = t_{\min}. \tag{3}$$

Let us show that the grouping LC'_l is optimal provided that the theorem assumptions are fulfilled, i.e., for given ε, p, and s,

$$T(p, LC_l, s, \varepsilon) = T(p, LC'_l, s, \varepsilon). \tag{4}$$

When $s = p$, for the optimal grouping LC_l, we have $t_{\max}(LC_l) = t_{\min}$ given. Therefore, the assertion of the theorem is valid.

Let $s > p$, $s = kp + r$, $1 \le r \le p$. Let us consider all possible cases.

(1) $T^n_\varepsilon \le p(t_{\min} + \varepsilon)$, $T^n_\varepsilon \le p(t_{\max}(LC_l) + \varepsilon)$. Then, from (1) and the optimality condition of LC_l, it follows that

$$\begin{gathered} T(p, LC'_l, s, \varepsilon) - T(p, LC_l, s, \varepsilon) \\ = T^n_\varepsilon + (s-1)(t_{\min} + \varepsilon) - T^n_\varepsilon - (s-1)(t_{\max}(LC_l) + \varepsilon) \\ = (s-1)(t_{\min} - t_{\max}(LC_l)) \ge 0. \end{gathered}$$

Since $s \ge 2$, from (3), it follows that $t_{\max}(LC_l) = t_{\min}$, and (4) holds.

(2) $T^s_\tau > p(t_{\min} + \varepsilon)$, $T^s_\tau > p(t_{\max}(LC_l) + \varepsilon)$. From (2), it follows that

$$\begin{gathered} T(p, LC'_l, s, \varepsilon) - T(p, LC_l, s, \varepsilon) \\ = (k+1)T^n_\varepsilon + (r-1)(t_{\min} + \varepsilon) \\ -(k+1)T^n_\varepsilon - (r-1)(t_{\max}(LC_l) + \varepsilon) \\ = (r-1)(t_{\min} - t_{\max}(LC_l)) \ge 0. \end{gathered}$$

Hence, by virtue of (3), (4) is valid.

(3) $T^n_\varepsilon > p(t_{\min} + \varepsilon) T^n_\varepsilon \le p(t_{\max}(LC_l) + \varepsilon)$. Then,

$$\begin{gathered} T(p, LC'_l, s, \varepsilon) - T(p, LC_l, s, \varepsilon) \\ = (k+1)T^n_\varepsilon + (r-1)(t_{\min} + \varepsilon) \\ - T^n_\varepsilon - (s-1)(t_{\max}(LC_l) + \varepsilon) \\ = (k+1)T^n_\varepsilon + (r-1)(t_{\min} + \varepsilon) \\ -(kp + r - 1)(t_{\max}(LC_l) + \varepsilon) \\ = kT^n_\varepsilon - kpt_{\max}(LC_l) + (r-1)(t_{\min} + \varepsilon) \\ -(r-1)(t_{\max}(LC_l) + \varepsilon) \\ = k(T^s_\tau - pt_{\max}(LC_l)) \\ +(r-1)(t_{\min} - t_{\max}(LC_l)) \ge 0, \end{gathered}$$

which is possible only for $r = 1$ and $T^n_\varepsilon = p(t_{\max}(LC_l) + \varepsilon)$. Hence, it follows that (4) is valid.

The theorem is proved.

Theorem 2. If $t_{\max}(LC_l) = t_{\max}(LC_{i-1})$ for groupings LC_l and LC_{l-1}, $l > 2$, then

$$T(p, LC_l, s, \varepsilon) > T(p, LC_{l-1}, s, \varepsilon).$$

From Theorem 1, it follows that, if one can effectively construct a linear grouping LC_l of blocks of identically distributed systems for each rank $l = 2, \ldots, n$ with the least maximum element among groupings of this rank ($t_{\max}(LC_l) = t_{\min}$), then the original problem will be effectively solved, since the optimal grouping in this case is to be chosen from $(n-1)$ groupings.

It is also obvious that the least maximum element among groupings of rank l does not decrease as l decreases, that is,

$$t_{\min}(LC_{l_1}) \ge t_{\min}(LC_{l_2}), \quad 1 < l_1 < l_2 \le n, \tag{5}$$

which allows one not to consider grouping into a single program block when solving the optimal grouping problem.

From the practical standpoint, it is reasonable to assume that

$$\varepsilon \le t_i, \quad i = \overline{1, n}, \tag{6}$$

which also allows one not to consider grouping into a single program block when solving the optimal grouping problem.

Along with the original problem, we consider the following optimization problem of the linear stowage into containers.

For given items of a finite ordered set $M = \{m_1, m_2, \ldots, m_n\}$, the corresponding sequence of their sizes $v(m_1), v(m_2), \ldots, v(m_n)$, $v(m_i) > 0$, $i = \overline{1, n}$, and container capacity $B > 0$, $B \max\limits_{1 \le i \le n}\{v(m_i)\}$, it is required to find a linear stowage of the set M such that the size of each element of the stowage $v(M_l)$ does not exceed B and l takes the least value.

In the general case, i.e., when there is no condition of stowage linearity, this problem is NP-hard in the strong sense, since, for $v(m_i) \in (0, 1)$, $i = \overline{1, n}$, $B = 1$, we have a classical optimization problem of stowage into containers. The stowage linearity condition, which is related to the problem of optimal grouping blocks of identically distributed systems, considerably simplifies the solution.

The problem of the linear stowage into containers is effectively solved by the following last-fit (LF) algorithm:

(1) The first item m_1 is loaded into the first container, and the other items are loaded in ascending order of their numbers.

(2) The item m_i, $i = \overline{2, n}$ is loaded into the last container among the partially packed ones if the number of items placed into does not exceed $B - m_i$; otherwise, it is loaded into the next empty container.

Optimality of the linear stowage constructed by the LF algorithm is easily proved by reductio ad absurdum. The LF algorithm requires no more than $3n$ elementary operations and is a part of the algorithm for solving the original problem of the optimal grouping.

4. OPTIMAL GROUPING ALGORITHM

Let $P_n = (t_1, t_2, ..., t_n)$ be a sequence of times of executing each block q_i, $i = \overline{1, n}$ by all n processes, $n \geq 3$; $p \geq 2$ be the number of processors; and ε be the time characterizing system overheads, $\varepsilon \leq t_i$, $i = \overline{1, n}$.

The algorithm for constructing the optimal linear grouping of blocks consists of the following steps.

1. Construct an array of $\frac{n(n+1)}{2} - 1$ numbers x_{ij}, $i = \overline{2, n}$, $j = \overline{1, i}$ according to the rule

$$\begin{aligned} x_{nj} &= t_j, \quad j = \overline{1, n}, \\ x_{n-1,j} &= x_{nj} + t_{j+1}, \quad j = \overline{1, n-1}, \\ &\dots\dots\dots\dots \\ x_{n-k,j} &= x_{n-k+1,j} + t_{j+k}, \quad j = \overline{1, n-k}, \\ &\dots\dots\dots\dots \\ x_{2j} &= x_{3j} + t_{j+n-2}, \quad j = 1, 2. \end{aligned} \tag{7}$$

Here, the numbers $x_{ij} = t_j + t_{j+1} + ... + t_{j+n-i}$ are the periods of all possible linear groupings of blocks.

2. Sort x_{ij} in the ascending order, simultaneously removing redundant identical elements and elements $x_{ij} < \max\limits_{1 \leq j \leq n}\{t_j\}$, to obtain an ascending sequence $v_1 < v_2 < ... < v_k$ for which $v_1 \leq \max\limits_{1 \leq j \leq n}\{t_j\}$, $n - 1 \leq k < \frac{n(n+1)}{2} - 1$.

3. Set $T_0 = T(p, n, s, \varepsilon)$, $P_0 = P_n$, $l_0 = n$, $i = 1$.

4. Taking the capacity B equal to v_i, $i = \overline{1, k}$, apply the LF algorithm to the original set of identically distributed competing processes. Let l_i be the rank of the obtained block grouping $(q_1, q_2, ..., q_n)$.

5. If $l_i = l_{i-1}$, the obtained grouping LC_i is not taken into consideration. Calculate $i = i + 1$ and go to Step 4.

6. Calculate $T_i = T(p, LC_{l_i}, s, \varepsilon)$. If $T_i < T_0$, set $T_0 = T_i$, $P_0 = P_i(LC_{l_i})$; otherwise, T_0 and P_0 are not changed.

7. If $l_i > 2$, calculate $i = i + 1$ and go to Step 4; otherwise, $l_i = 2$. Stop.

After the completion of the algorithm, T_0 is equal to the minimum of functional (1), (2), and P_0 is the optimal grouping. Correctness of the algorithm follows from Theorems 1 and 2 and relations (4) and (5).

The above-described algorithm requires no more than $O(n^3)$ elementary operations, with the first stage involving $O(n^2)$ elementary operations to construct the array of x_{ij}, $i = \overline{2, n}$, $j = \overline{1, i}$, the second stage involving rapid sorting algorithms and $O(n^2 \log_2 n)$ operations, and the fourth stage involving at most $O(n^3)$ operations in v_i cycle.

i	j								
	1	2	3	4	5	6	7	8	
2	37	35							
3	32	32	33						
4	30	27	30	23					
5	25	25	25	20	20				
6	20	20	23	15	17	15			
7	17	15	18	13	12	12	10		
8	5	12	13	8	10	7	7	8	
9	5	2	10	3	5	5	2	5	3

Fig. 3.

Example. Let $P_9 = (5, 2, 10, 3, 5, 5, 2, 5, 3)$ be the sequence of execution times of blocks q_i, $i = \overline{1, 9}$; $p = 3$ be the number of processors; $s = 5$ be the number of blocks of a linearly structured program resource; and $\varepsilon = 1$ be system overheads on each block associated with the structuring and pipelining. 1

Since $5 = s > p = 3$ and the overall time of execution of each block Q_j by all n processes with regard to overheads ε is given by

$$\begin{aligned} T_\varepsilon^n &= \sum_{i=1}^{n} t_i^\varepsilon = \sum_{i=1}^{9} t_i + n\varepsilon \\ &= 40 + 9 = 49 > p t_{\max}^\varepsilon = 3(10 + 1) = 33, \end{aligned}$$

the overall execution time of $n = 9$ identically distributed processes that use the PR program resource structured into $s = 5$ blocks in the computing environment with $p = 3$ processors for asynchronous mode and the mode of continuous executing each block by all the processors according to formula (2) adds up to

$$\begin{aligned} T(3, 9, 5, 1) &= (k + 1) T_\varepsilon^n + (r - 1) t_{\max}^\varepsilon \\ &= 2 \cdot 49 + 1 \cdot 11 = 109 \text{ time units.} \end{aligned}$$

Let us find the linear grouping LC_l of the initial identically distributed system on which the minimum of functionals (1), (2) is attained using the above-described algorithm.

Table

B	$P_t(LC_{l_i})$	l_i	$T(p, LC_{l_i}, s, \varepsilon)$
10	7, 10, 8, 7, 8	5	$T_1^5 = 40 + 5 = 45 > 33 = 3(10 + 1) = pt_{10}^1$, then $T(3, 5, 5, 1) = (1 + 1)T_1^5 + (2 - 1)t_{10}^1 = 2 \times 45 + 1 \times 11 = 101$
12	7, 10, 8, 12, 3	5	–
13	7, 13, 12, 8	4	$T_1^4 = 40 + 4 = 44 > 42 = 3(13 + 1) = pt_{13}^1$, then $T(3, 4, 5, 1) = (1 + 1)T_1^4 + (2 - 1)t_{13}^1 = 2 \times 44 + 1 \times 14 = 102$
15	7, 13, 12, 8	4	–
17	17, 15, 8	3	$T_1^3 = 40 + 3 = 43 > 54 = 3(17 + 1) = pt_{17}^1$, then $T(3, 3, 5, 1) = T_1^3 + (5 - 1)t_{17}^1 = 43 + 4 \times 18 = 115$
20	20, 17, 3	3	–
23	20, 17, 3	3	–
25	25, 15	2	$T_1^2 = 40 + 2 = 42 > 78 = 3(25 + 1) = pt_{25}^1$, then $T(3, 2, 5, 1) = T_1^2 + (5 - 1)t_{25}^1 = 42 + 4 \times 26 = 146$

We construct the array of $\frac{9(9+1)}{2} - 1 = 44$ numbers of x_{ij}, $i = \overline{2, 9}$, $j = \overline{1, i}$ according to rule (7). Figure 3 shows the scheme of generating these numbers, which are sums of numbers at the bottom of the corresponding triangle.

By sorting x_{ij} in the ascending order and removing redundant identical elements and elements $x_{ij} < 10$, we obtain the following ascending sequence:

(10, 12, 13, 15, 17, 20, 23, 25, 30, 32, 33, 35, 37). (8)

Assuming that the capacity B successively takes values of elements in sequence (8), we apply the LF algorithm to the initial system of identically distributed competing processes until it gives the grouping of rank 2. Calculations performed at each step are presented in the table, where the dashes mean that there is no need to calculate the corresponding values of $T(p, LC_i, s, \varepsilon)$.

As can be seen from the table, the optimal grouping is given by

$$LC_5 = (Q_1 \cup Q_2, Q_3, Q_4 \cup Q_5, Q_6 \cup Q_7, Q_8 \cup Q_9),$$

for which $P_t(LC_5) = (7, 10, 8, 7, 8)$ and $T(3, 5, 5, 1) = 101$.

Thus, the overall execution time of the initial identically distributed system is improved through the optimal grouping by $109 - 101 = 8$ time units, or by $\frac{8}{109} \times 100\% \approx 7.34\%$.

Correctness of the algorithm operation is supported by results obtained in [5, 6], where it was shown that the optimal identically distributed system is to be found among stationary identically distributed systems. The following theorem is proved.

Theorem. In order that an efficient identically distributed system of competing processes in the case of the limited parallelism in the asynchronous and the second synchronous modes be optimal for given $p \geq 2$, T_ε^n, $\varepsilon > 0$, it is necessary and sufficient that the system is stationary and the number of processes n_0 is equal to one of the following numbers:

(1) $\left[\left[\sqrt{\frac{(p-1)T_\varepsilon^n}{k\varepsilon}}\right], \left[\sqrt{\frac{(p-1)T_\varepsilon^n}{k\varepsilon}}\right] + 1\right] \cap [2, n]$, for $s = kp$, $k > 1$,

(2) $\left[\left[\sqrt{\frac{(r-1)T_\varepsilon^n}{(k+1)\varepsilon}}\right], \left[\sqrt{\frac{(r-1)T_\varepsilon^n}{(k+1)\varepsilon}}\right] + 1\right] \cap [2, n]$, for $s = kp + r$, $k \geq 1$, $1 \leq r < p$,

where the function $\bar{\Delta}_\varepsilon(x) = (s-1)T^n\left(1 - \frac{1}{x}\right) - (x + s - 1)\varepsilon$, $x \geq 1$ reaches its maximum, $[z]$ denotes the greatest integer not exceeding z, and n is a given number.

For the above example, the number of processes is

$$n_0 = \left[\sqrt{\frac{(r-1)T_\varepsilon^n}{(k+1)\varepsilon}}\right] + 1 = \left[\sqrt{\frac{40}{2}}\right] + 1 = [4, 47] + 1 = 5,$$

which substantiates the rank of the optimal grouping obtained.

REFERENCES

1. Voevodin, V.V. and Voevodin, Vl.V., *Parallel'nye vychisleniya* (Parallel Computing), St. Petersburg: BKhV–Peterburg, 2002.
2. Kovalenko, N.S. and Samal', S.A., *Vychislitel'nye metody realizatsii intellektual'nykh modelei slozhnykh sistem* (Computational Methods of Implementation of Intelligent Models of Complex Systems), Minsk: Belaruskaya navuka, 2004.
3. Toporkov, V.V., *Modeli raspredelennykh vychislenii* (Distributed Computing Models), Moscow: Fizmatlit, 2004.
4. Ivannikov, V.P., Kovalenko, N.S., and Metel'skii, V.M., On the Minimal Time Required for Execution of Distributed Concurrent Processes in Synchronous Modes, *Programming Comput. Software*, 2000, vol. 26, no. 5, pp. 268–274.
5. Kapitonova, Yu.V., Kovalenko, N.S., and Pavlov, P.A., Optimality of Systems of Equally Distributed Competing Processes, *Kibern. sistemnyi analiz,* 2005, no. 6, pp. 3–10.
6. Pavlov, P.A., Efficiency of Distributed Computing in Scalable Systems, *Nauchno-tekhnicheskie vedomosti SPbGPU*, 2010, no. 1, pp. 83–89.

SPELL: 1. pipelining