Analysis of Exchanging Interactions in Some Parallel Algorithms of the Linear Equational System

N. Vasilev

Key Words: *Parallel algorithm; data exchange; exchanging interactions.*

Abstract. *This paper analyses and minimizes the exchanging interactions in the parallel algorithm of type „allocation of submatrices with identical dimensions in the processors of computer system with parallel architecture" of the linear equational system, solved by the method of sequential interations.*

1. Introduction

An effective method for accelerating the solving a given task is to use a solution based on parallel algorithms (PA). There is a growing interest in searching and finding PA for solving complex problems in the last years [1,2,5]. PA presents the task as a set of simultaneously executed subtasks which exchange data throughout the process of solving. It is preferable these data to be minimal [3,4], because in this way the traffic in the communicational network is decreased and, in the common case, the control of the parallel process is facilitated.

2. Aims

The aim of this article is: to analyze and minimize the exchanging interactions in PA of type „allocation of submatrices with identical dimensions in the processors of computer system (CS) with parallel architecture" of the linear equational system, solved by the method of sequential interations. (The essence of this PA is presented below. It can be referred to the so-called parallel matrix computations with checkerboard partitioning [1].) In details, the aim of this paper (of the presented task) will be:
1) determining the number of transmitted and received words between the threads (the branches) of the PA;
2) analysing the possible allocations of the matrix elements **α** (see pt. 3) and suggesting allocations, in which the algorithms for routing of the exchanged words are simple and the number of the words – transmitted to and received in the parallel branches is minimal;
3) analysing the dependence of the number of words - transmitted to and received in the parallel branches on the submatrix dimension, aiming the determination of dimension, in which this number is minimal.
The Solution of the problems, presented above, will be searched considering the following conditions:
1) the matrices of the linear equational system do not contain zeros;
2) the congruity of the solution is acceptable.
This paper has an even further aim: searching forethfor analysis of the exchanging interactions in PA of complex tasks for finding optimal PA (according to defined criteria) and optimal topologies for communication between the parallel branches of the PA.

3. Method of Sequential Iterations for Solving the Linear Equational System

Let us have the linear equational system:

$$\begin{aligned}
&a_{11}\, x_1 + a_{12}\, x_2 + \ldots + a_{1n}\, x_n = b_1;\\
&a_{21}\, x_1 + a_{22}\, x_2 + \ldots + a_{2n}\, x_n = b_2;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n1}\, x_1 + a_{n2}\, x_2 + \ldots + a_{nn}\, x_n = b_n.
\end{aligned}$$

In matrix form: **AX=B**, where

$$\mathbf{A}=\begin{vmatrix} a_{11} & a_{12} & \ldots & a_{1n}\\ a_{21} & a_{22} & \ldots & a_{2n}\\ \ldots & \ldots & \ldots & \ldots\\ a_{n1} & a_{n2} & \ldots & a_{nn}\end{vmatrix};\quad X=\begin{vmatrix} x_1\\ x_2\\ \ldots\\ x_n\end{vmatrix};\quad \mathbf{B}=\begin{vmatrix} b_1\\ b_2\\ \ldots\\ b_n\end{vmatrix}.$$

If $\alpha_{ij} \neq 0,\ i,=1,2,\ldots n,$ then

$$\begin{aligned}
&x_1=\beta_1+\alpha_{11}\, x_1+\alpha_{12}\, x_2+\ldots+\alpha_{1n}\, x_n;\\
&x_2=\beta_2+\alpha_{21}\, x_1+\alpha_{22}\, x_2+\ldots+\alpha_{2n}\, x_n;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&x_n=\beta_1+\alpha_{n1}\, x_1+\alpha_{n2}\, x_2+\ldots+\alpha_{nn}\, x_n;
\end{aligned}$$

$$\beta_i=\frac{bi}{aii},\ \alpha_{ij}=-\frac{aij}{aii}\ if\ i\neq j,\ (\alpha_{ij}\neq 0),$$

$$\alpha_{ij}=0\ if\ i=j,\quad i,j=1,2,\ \ldots n.$$

In matrix form: $X=\beta+\alpha X$
The column of free members is considered for zero approximation: $X^{(0)}=\beta$. After that, the matrix-columns are computed $X^{(1)}=\beta+\alpha X^{(0)}$ (first approximation), $X^{(2)}=\beta+\alpha X^{(1)}$ (second approximation) and so on. Each approximation is calculated according to the formula $X^{(\kappa+1)}=\beta+\alpha X^{(\kappa)}$, $\kappa=1,2,\ldots$.
If the order $X^{(0)}, X^{(1)}, X^{(2)},\ldots X^{(\kappa)},\ldots$ has a limit $X=\lim_{k\to\infty} X^{(\kappa)}$, this limit is solution of the system. Practically, when finding an approximation $X_i^{(\kappa)}$ $i = 1, 2, \ldots n$ the difference is calculated

$\Delta X_i^{(k)} = X_i^{(k)} - X_i^{(k-1)}$. The task is considered solved when the condition $\Delta X^{(k)} < \varepsilon$ is satisfied, where $i = 1, 2, \dots n$ and $\varepsilon > 0$. ε is a given small number, defining the calculation precision.

Solving the task includes $\mathbf{n^2}$ multiplications, $\mathbf{n^2}$ sums, **n** subtractions and **n** comparisons in one iteration cycle. It can be seen that the great part of computing work includes multiplications (n^2) and summings (n^2).

4.Exchanging Interactions in PA of Type "Checkerboard Partitioning" of the Linear Equational System, Solved by the Method of Sequential Iterations

Let the number of processors be **m**. The contemporary CS with parallel architecture are built of dozens, hundreds and, sometimes, even thousands of processor elements [1]. In this paper, it will be considered that **m=n**. It will be shown this does not influence the generality of the considerations.

In the checkerboard partitioning, the matrix α is split in **n** submatrices with identical dimensions (or dimensions that differ with 1). For facilitating the presentation, it is assumed that the submatrices are identical. Let the number of rows of the submatrix be **r**, and the number of columns be **c**. It is not difficult to see that **rc=n**. The number of the submatrices in each row of the checkerboard partitioning is **r**, and their number in each column is **c**. When **r=1** the partitioning is block-rowed (**1** x **n**), and when **c=1** – it is block columned (**1** x **n**)[1].

The values of each submatrix are allocated (saved) in one of the processors of CS. Each processor calculates one x_i, **i=1,2,...,n.** The Processor P_i will be considered to calculate x_i. It is obvious that value of β_i has to be saved in the processor P_i. The allocation of submatrices with dimension 2x3 in processors of CS (**n=6**) is shown in *figure 1*.

The number of possible allocations of submatrices in the processors is **n!**. The allocation of submatrices, itself, defines the quantity of transmitted and received words between the parallel branches of PA and the loading of the communicational network.

$\underline{P_1 - x_1}$	a_{11},a_{12},a_{13}	$\underline{P_2 - x_2}$	a_{14},a_{15},a_{16}
b_1	a_{21},a_{22},a_{23}	b_2	a_{24},a_{25},a_{26}
$\underline{P_3 - x_3}$	a_{31},a_{32},a_{33}	$\underline{P_4 - x_4}$	a_{34},a_{35},a_{36}
b_3	a_{41},a_{42},a_{43}	b_4	a_{44},a_{45},a_{46}
$\underline{P_5 - x_5}$	a_{51},a_{52},a_{53}	$\underline{P_6 - x_6}$	a_{54},a_{55},a_{56}
b_5	a_{61},a_{62},a_{63}	b_6	a_{64},a_{65},a_{66}

Figure 1. **P**-type allocation of submatrices (dimension 2x3, n=6)

Each processor P_i in CS:

1)calculates **n** multiplications;

2)executes **(c-1)r** summings and, in result, **r** partial sums are calculated;

3)transmits **r** or **r-1** partial sums respectively to **r** or **r-1** processors, and receives **r** or **r-1** partial sums respectively from **r** or **r-1** processors;

4)executes **r-1** summings of the **r** partial sums in the processor and, in result, x_i is calculated;

5)if the parallel process is over - **end**, otherwise the 6-th step of the algorithm is executed for the completion of the next iteration;

6)x_i is transmitted to **c** or **c-1** processors, receives **c** or **c-1** x_{kj}, **j=1,2,...,c-1** or **j=1,2,...,c; kj** $\in$ **{1,2,..., i-1,i+1,...,n}** from **c** or **c-1** processors respectively and goes on with the 1-st step of the algorithm.

The control of the calculation process (the 5-th step) is not a point to be discussed in this article. It is presented in [3,4].

More attention will be paid to the 3-rd and the 6-th step, which define the exchanging interactions between the PA branches, i.e. between the processors of CS.

Each processor P_i has to transmit **r** or **r-1** partial sums respectively to **r** or **r-1** processors, and to receive **r** or **r-1** partial sums respectively from **r** or **r-1** processors (the 3-rd step). It is obvious that the case transmit/receive **r-1** partial sums is preferable. This case is an allocation of submatrices, when one of the calculated **r** partial sums is in the P_i processor and participates in the calculation of x_i. Therefore, such an allocation of submatrices has to be done that each processor in the CS P_i, **i=1,2,...,n** calculates one of the partial sums necessary for the computation of its x_i. Such allocation will be called **P-type** allocation.

Rule 1. A **P**-type allocation is achieved when the index of each processor P_i, computing x_i, **i=1,2,...,n,** coincides with one of the **r** row-numbers of the submatrix, allocated in this processor. In other words, each of the **r** processors in each row of the checkerboard partitioning computes one of these x_i, whose indices are the **r** row-numbers of the submatrices, allocated in the **r** processors.

The allocation in *figure 1* complies with this condition. The number of all such allocations is $\mathbf{(r!)^c}$.

In this case the number of the transmitted S_p and received R_p words (partial sums) to and from each processor P_i through the communicational network is:

$$S_p = R_p = r\text{-}1.$$

The total number of transmitted S_{pt} and received R_{pt} words (partial sums) to and from all processors through the communicational network is:

$$S_{pt} = R_{pt} = n(r\text{-}1).$$

Attention should be poid to the fact that through the exchange of partial sums the set of processors **n** is split to **c** independent subsets, each of them containing **r** processors (these in the rows of the checkerboard partitioning). This makes the routing algorithm simpler.

After the **k**-th approximation of x_i, **i=1,2,...,n** is calculated (the 4-th step), if the parallel computing process has not finished, a new iteration cycle starts. For this purpose, the CS processors have to exchange the **k**-th approximations of x_i. Each processor P_i has to transmit the calculated by it **k**-th approximation of x_i, to **c** or **c-1** processors and has to receive **c** or **c-1** values of the **k**-th approximations of x_i (these that are necessary for the calculation of the **n** multiplications, defined from the submatrix in the processor) from **c** or **c-1** processors respectively (the 6-th step). The case transmit one value to **c-1** processors and receive of **c-1** values of the **k**-th approximations of

x_i is preferable. This case finds its place when the **k**-th approximation of x_i is used in processor P_i for the calculation of **r** from the **n** multiplications of the **(к+1)-st** approximation of x_i. Therefore, such an allocation of the submatrices has to be made that the above-stated condition is complied with all the processors of CS. Such allocation will be called **I**-type allocation.

Rule 2. An **I**-type allocation is achieved when the index of each processor P_i, computing x_i, **i=1,2,...,n,** coincides with one of the **c** column-numbers of the submatrix that is allocated in this processor. In other words, each of the **c** processors in each column of the checkerboard partitioning computes one of these x_i, whose indices are the **c** column-numbers of the submatrices, allocated in the **c** processors.

The allocation in *figure 2* complies with this condition. The number of all such allocations is **(c!)**^{r}.

	$x_1\ x_2\ x_3$		$x_4\ x_5\ x_6$
$\underline{P_1 - x_1}$,	a_{11},a_{12},a_{13}	$\underline{P_4 - x_4}$,	a_{14},a_{15},a_{16}
b_1	a_{21},a_{22},a_{23}	b_4	a_{24},a_{25},a_{26}
$\underline{P_2 - x_2}$,	a_{31},a_{32},a_{33}	$\underline{P_5 - x_5}$,	a_{34},a_{35},a_{36}
b_2	a_{41},a_{42},a_{43}	b_5	a_{44},a_{45},a_{46}
$\underline{P_3 - x_3}$,	a_{51},a_{52},a_{53}	$\underline{P_6 - x_6}$,	a_{54},a_{55},a_{56}
b_3	a_{61},a_{62},a_{63}	b_6	a_{64},a_{65},a_{66}

Figure 2. **I**-type allocation of submatrices (dimension 2x3, n=6)

In this case, the number of the transmitted S_i and received R_i words (unknowns x_i) to and from each processor P_i through the communicational network is:

$$S_i = 1 \text{ (to c-1 processors)}; \; R_i = c\text{-}1.$$

The total number of transmitted S_{it} and received R_{it} words (unknowns x_i) to and from all processors through the communicational network is:

$$S_{it} = n; \; R_{it} = n(c\text{-}1).$$

One should to the fact that through the exchange of the unknowns x_i the set of processors **n** is split to **r** independent subsets, each of them containing **c** processors (these in the columns of the checkerboard partitioning). This makes the routing algorithm simpler.

The allocation *in figure1* is not optimal regarding the exchange of the unknowns x_i. The processors P_2 and P_5 have to transmit the calculated by themselves values of x_2 and x_5, to **c**(3) processors: P_2 to P_1, P_3 and P_5, and P_5 to P_2, P_4 and P_6, and have to receive **c**(3) values of unknowns: P_2 - x_4, x_5 and x_6, and P_5 - x_1, x_2 and x_3.

The allocation in *figure 2* is not optimal regarding the exchange of the partial sums. The processors P_2, P_3, P_4 and P_5 have to transmit partial sums to **r**(2) processors and have to receive partial sums from **r**(2) processors: P_2 transmits to P_3 and P_4, receives from P_1 and P_4; P_3 transmits to P_5 and P_6, receives from P_2 and P_5; P_4 transmits to P_1 and P_2, receives from P_2 and P_5; P_5 transmits to P_3 and P_4, receives from P_3 and P_6.

5. PA of Type „Allocation of Submatrices with Identical Dimensions of Rearranged by Rows and Columns Matrix“

A question arises: could it not be found an allocation of submatrices that would comply with both rules? If it could, that would be the best allocation because:

- the number of exchanged words would be minimal;
- the routing algorithms would be simpler.

Applying both rules simultaneously in increasing order of the row numbers and column numbers is apparently impossible, because the solutions are mutually exclusive. This can be done only if these numbers are rearranged. In processor P_i, computing x_i, there really have to be a submatrix, containing the number of a row and a column of the matrix, coinciding with the index **i**. In other words, the indices of the processors on a given row (column) in the checkerboard partitioning define the numbers of the rows (columns) of the submatrices, allocated in the processors on the row (column) of the checkerboard partitioning. The aforementioned defines the rules for finding the submatrices at a given checkerboard partitioning, i.e. the rules for rearranging of the rows and columns of the matrix. In this case „a submatrix“ does not include only sequential rows and columns.

Column / Row	1 2 3 4 12 7 1 5	5 6 7 8 2 3 9 4	9 10 11 12 6 10 8 11
1 12 2 2 3 6	P_{12} - X_{12}	P_2 - X_2	P_6 - X_6
4 7 5 3 6 10	P_7 - X_7	P_3 - X_3	P_{10} - X_{10}
7 1 8 9 9 8	P_1 - X_1	P_9 - X_9	P_8 - X_8
10 5 11 4 12 11	P_5 - X_5	P_4 - X_4	P_{11} - X_{11}

Figure 3. **P**-type and **I**-type allocation of submatrices (dimension 3x4, n=12)

A rearrangement of rows and columns of the matrix **(n=12)**, is shown in *figure 3*. The proposed allocation of submatrices with dimension **(3x4)** between the processors of CS complies with both rules. It can be seen that the numbers of the rows (columns) of the matrix in each row (column) of the checkerboard partitioning are identical with the indices of the processors in this row (column). For example, they are 7, 3, 10 in the second row, and 12, 7, 1, 5 in the first column. The order of the row (column) numbers of the matrix in the rows (columns) of the checkerboard partitioning does not matter. The number of the possible rearrangements of a given checkerboard partitioning is **(r!)**^{c}**(c!)**^{r}.

Let us look at the work of one of the processors - P_9. It computes the following three partial sums:

$$\alpha_{12}x_2+\alpha_{13}x_3+\alpha_{19}x_9+\alpha_{14}x_4$$
$$\alpha_{92}x_2+\alpha_{93}x_3+\alpha_{99}x_9+\alpha_{94}x_4$$
$$\alpha_{82}x_2+\alpha_{83}x_3+\alpha_{89}x_9+\alpha_{84}x_4$$

The processor **P_9** transmits the first partial sum to **P_1**, and the third one to **P_8**, i.e. to the processors that are in its row of the checkerboard partitioning. The second partial sum stays in processor **P_9**. It is necessary for the calculation of **x_9**. The other two partial sums, necessary for the computation of **x_9**:

$$\alpha_{9,12}x_{12}+\alpha_{97}x_7+\alpha_{91}x_1+\alpha_{95}x_5$$
$$\alpha_{96}x_6+\alpha_{9,10}x_{10}+\alpha_{98}x_8+\alpha_{9,11}x_{11}$$

The processor **P_9** receives from the processors **P_1** and **P_8** respectively. When **x_9** is calculated, the processor **P_9** transmits its value to processors **P_2**, **P_3** and **P_4**, i.e. to the processors that are in its column of the checkerboard partitioning.

The other 11 processors have the same work organization.

row \ column	1 2 3 4 1 4 7 10	5 6 7 8 2 5 8 11	9 10 11 12 3 6 9 12
1 2 3	P_1 - X_1	P_2 - X_2	P_3 - X_3
4 5 6	P_4 - X_4	P_5 - X_5	P_6 - X_6
7 8 9	P_7 - X_7	P_8 - X_8	P_9 - X_9
10 11 12	P_{10} - X_{10}	P_{11} - X_{11}	P_{12} - X_{12}

Figure 4. **P**-type and **I**-type allocation of submatrices with rearrangement only of the column numbers of the matrix (dimension 3x4, n=12)

The allocation in *figure 3* is incidentally chosen in order to illustrate the rearrangement of the numbers of rows and columns of the matrix. The most probable rearrangements are shown in *figure 4* and *figure 5*.

The allocation in *figure 4* complies with the conditions of Rule 1. That is why, in this case, only a rearrangement of the column numbers of the matrix is necessary.

The allocation in *figure 5* complies with the conditions of Rule 2. That is why, in this case, only a rearrangement of the row numbers of the matrix is necessary.

row \ column		1 2 3 4	5 6 7 8	9 10 11 12
1 2 3	1 5 9	P_1 - X_1	P_5 - X_5	P_9 - X_9
4 5 6	2 6 10	P_2 - X_2	P_6 - X_6	P_{10} - X_{10}
7 8 9	3 7 11	P_3 - X_3	P_7 - X_7	P_{11} - X_{11}
10 11 12	5 4 11	P_4 - X_4	P_8 - X_8	P_{12} - X_{12}

Figure 5. **P**-type and **I**-type allocation of submatrices with rearrangement only of the row numbers of the matrix (dimension 3x4, n=12)

It can be seen that the order of the rearranged numbers of the rows/columns is cyclic. For the rows (*figure 5*) the cycle is 4(**c**), and for the columns (*figure 4*) – 3(**r**).

An incidental allocation of the submatrices without rearrangement complicates the routing algorithms and the number of exchanged words in PA is not minimal.

6. Dependence of the Number of the Exchanged Words on the Dimension of the Submatrices of Allocation

The dependences of the number of the exchanged words for one branch of PA and for all branches of PA, go in pt.4, are shown in *table 1*.

A processor (branch) transmits least number of words when **r**=1, and it receives least number of words when **$r=n^{1/2}$**. Practically, this means that **r** and **c** are necessary to be close by value in order a minimal number of received words to be achieved. For example, if **n**=900, the processors will receive minimal number of words – 58, when **r**=**c**=30. If **n**=800, because $800^{1/2} \approx 28{,}3$, i.e. it is not an integer, one good choice is **r**=25, **c**=32. In this case the number of received words is 55.

The number of exchanged words (received and transmitted) is minimal when **$r=(n/2)^{1/2}$**. Therefore, **$c=(2n)^{1/2}$**, i.e. **c**/**r**=**2**. For example, if **n**=800, the best case is **r**=20, **c**=40. The number of exchanged words from each processor (branch) will be 78, 20 of which will be transmitted and 58 - received.

Table 1. Dependencies of the number of the exchanged words on the dimension of the submatrices

	For one branch of PA		For all branches of PA	
	Transmitted words **S**	Received words **R**	Transmitted words **S**	Received words **R**
Partial sums	$S_p = r-1$	$R_p = r-1$	$S_{pt} = n(r-1)$	$R_{pt} = n(r-1)$
Unknowns x_i	$S_i = 1$	$R_i = c-1 = n/r-1$	$S_{it} = n$	$R_{it} = n(n/r-1)$
Total	$S = r$	$R = r+n/r-2$	$S_t = nr$	$R_t = n(r+n/r-2)$
Total exchanged words	$S+R = 2r+n/r-2$		$S_t+R_t = n(2r+n/r-2)$	

The block-row partitioning and, especially, the block-column partitioning are the most unfavourable — the quantity of exchanged words is maximal (see *figure 6*) and each processor has to exchange (transmit and receive) words with each one of the other processors.

The dependences of the number of exchanged words on the number of the rows (**r**) of the matrix of allocation for one branch of PA when **n=16** is shown in *figure 6*.

7. Avoiding the Restrictions in the Proposed PA

At the end, the two restrictions will be discussed:

— the number of the processors is equal to the number of the unknowns x_i, i.e. **m=n** (**m,n** — integers);

— the dimension of the submatrices of allocation is identical.

The removing of the first restriction puts ahead the question for the following dependences.

1. m>n. This case is not very probable. (Solving tasks with PA makes sense only when they involve a great number of operations, i.e. they are tasks with great dimensions.) The execution of PA of the task will engage **n** of the **m** processors, i.e. this will be the aforementioned case.

2. n>>m. It will be considered that **n/m = t** is an integer. (This condition is connected with the second restriction, which will be discussed below.) In this case, each processor will compute the values of **t** unknowns x_i. Therefore,

$$rc = tn;\ r,c \geq t$$

In comparison with the case when **m=n**, the quantity of work for each processor increases **t** times.

The dependences of the number of exchanged words for one branch of PA and for all branches of PA is shown in *table 2*.

The number of exchanged words is minimal when $r=(tn/2)^{1/2}$.

3. n>m>n/2. This time it is appropriate PA of the task to be executed by m_1 of the **m** processors (m_1<**m**), where $m_1 \leq n/2$. There is maximal acceleration when $m_1 = n/2$. (It will be considered that $n/(2m_1)$ is an integer.)

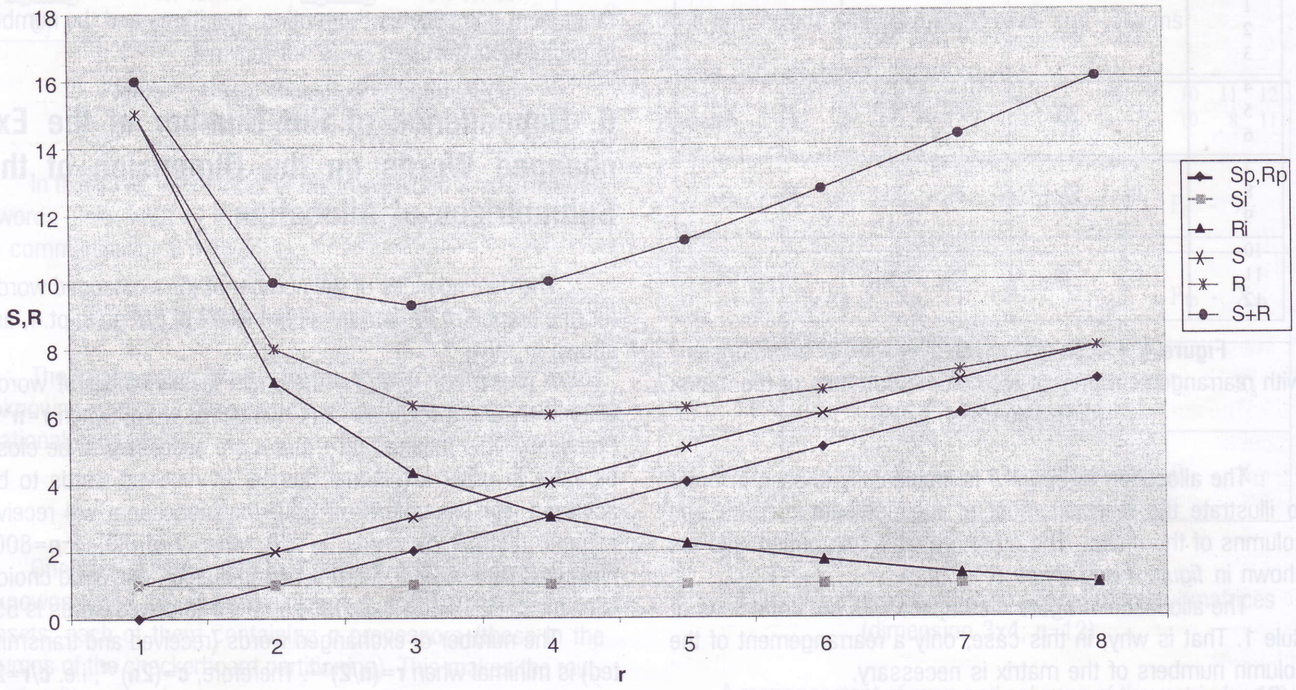


Figure 6. Dependencies of the number of exchanged words on the number of the matrix rows (**r**) for one branch of PA (n=16)

Table 2. Dependencies of the number of the exchanged words on the dimension of the submatrices (n>>m, n/m=t - integer)

	For one branch of PA		For all branches of PA	
	Transmitted words **S**	Received words **R**	Transmitted words **S**	Received words **R**
Partial sums	$S_p = r\text{-}t$	$R_p = r\text{-}t$	$S_{pt} = m(r\text{-}t)$	$R_{pt} = m(r\text{-}t)$
Unknowns x_i	$S_i = t$	$R_i = c\text{-}t = tn/r\text{-}t$	$S_{it} = mt = n$	$R_{it} = m(tn/r\text{-}t)$
Total	$S = r$	$R = r+tn/r\text{-}2t$	$S_t = mr$	$R_t = m(r+tn/r\text{-}2t)$
Total exchanged words	$S+R = 2r+tn/r\text{-}2t$		$S_t + R_t = m(2r+tn/r\text{-}2t)$	

The removing of the second restriction puts ahead the need of the task analysis, which will be done in another article. Apparently, the restriction for identity of the dimensions of the submatrices can not be fulfilled by any **n** (for example, when **n** is a simple number), and in some cases it is far from the optimal solution (for example, when **n=2k**, where **k** is a simple number).

The avoiding of the restriction (still keeping the advantages – simplicity of the routing algorithms and minimal word exchange) is based on the possibility for each **n**, when $(n/2)^{1/2}$ is not an integer, an allocation with rearrangement to be made, including submatrices with 4 dimenions: **rxc**, **(r+1)xc**, **rx(c+1)** and **(r+1)x(c+1)**. The row (column) number of the submatrices in the rows (columns) of the checkerboard partitioning has to be identical.

Let **n** be given, where $(n/2)^{1/2}$ is not an integer. The number of the processors **m** and the dimension of the submatrices, i.e. **r** and **c,** have to be defined. Using the condition for minimal exchange, $r=](n/2)^{1/2}[$ is computed. (The sign][is for the nearest smaller integer.) The integers n_1, **c** and n_2 are computed using respectively the dependences $r=(n_1/2)^{1/2}$, $rc=n_1$ and $r+1=(n_2/2)^{1/2}$. Apparently, $n_1<n<n_2$. When $n=n_1$, optimal solution is achieved if the dimension of the submatrices is **rxc**, and when $n=n_2$ – if the dimension is **(r+1)x(c+2)**. $n_{12}=(r+1)c=r(c+2)$ is computed. It is clear that $n_1<n_{12}<n_2$.

Now the number of the processors **m** can be calculated. When $n<n_{12}$, $m=n_1$. When $n>n_{12}$, $m=n_2$. When $n=n_{12}$, $m=n_{12}$.

In the first case, $q=n-n_1$ of the n_1 processors ($q<2r$) have to compute one more x_i, i.e. two unknowns x_i. This means that in **q** rows and **q** columns of the checkerboard partitioning the number of the submatrix rows of the checkerboard partitioning will increase with 1, and the number of the submatrix columns – with 1 when $q \leq r$ and with 2 (when $q>r+1$) in comparison with submatrix of n_2. Let the numbering of the unknowns in the checkerboard partitioning be sequential, horizontal. This means that only the columns of the matrix have to be rearranged. The processors (**q** in number), that will compute two unknowns x_i, have to be one in a row and one (when $q \leq r$) or two (when $q>r$) in a column.

In the second case, $q=n_2-n$ of the n_2 processors ($q<2(r+1)$) do not have to compute x_i. This means that in **q** rows and **q** columns of the checkerboard partitioning the number of the submatrix rows will decrease with 1, and the number of the submatrix columns – with 1 (when $q \leq r+1$) and with 2 (when $q>r+1$) in comparison with the submatrix of n_2. Let the numbering of the unknowns in the checkerboard partitioning be sequential, horizontal. This means that only the columns of the matrix have to be rearranged. The processors (**q** in number), that will not compute x_i, have to be one in a row and one (when $q \leq r+1$) or two (when $q>r+1$) in a column.

The difference in the computing work of the processors at such an allocation will be as follows. The processors with dimension of the submatrices **(r+1)xc** calculate **c** multiplications and **c-1** sums more than the processors with dimension of the submatrices **rxc**. The processors with dimension of the submatrices **rx(c+1)** – **r** multiplications and **r** sums more. The processors with dimension of the submatrices **(r+1)x(c+1)** – **r+c-1** multiplications and sums **r+c** more. With the increasing of **n** the relative differences in the computation work of the processors will decrease. There is difference in the computation work of the processors also because of the differences in the number of the computed x_i. For each of them **r-1** sums are calculated.

The dependences of the number of exchanged words for one branch of PA in *table 1* and *table 2* are actual for the proposed allocation. However, the number of exchanged words in PA is not this one. It is the sum of the exchanged words in all the parallel branches and, in this case, not all submatrices have identical **r** and **c**.

Another allocations can be made, of course, but in the common case, they will not be optimal.

The allocation for **n=11** is shown in *figure 7*, and for **n=14** – in *figure 8*. For both cases $n_1=8$, $n_2=18$, $n_{12}=12$. When **n=11**, $n<n_{12}$. That is why $m=n_1=8$. The processors P_4, P_5 and P_8 (**q=3**) compute two unknowns x_i. When **n=14**, $n>n_{12}$. That is why $m=n_2=18$. The processors P_9, P_{10}, P_{14} and P_{18} (**q=4**) do not compute x_i.

Column Row	1 2 3 4 5 1 3 6 7 9	6 7 8 9 10 11 2 4 5 8 10 11
1, 2	P_1 - X_1	P_2 - X_2
3, 4, 5	P_3 - X_3	P_4 - X_4, X_5
6, 7, 8	P_5 - X_6, X_7	P_6 - X_8
9, 10, 11	P_7 - X_9	P_8 - X_{10}, X_{11}

Figure 7. Allocation of submatrices for n=11

Column Row	1 2 3 4 5 1 4 7 11 13	6 7 8 9 10 2 5 8 9 14	11 12 13 14 3 6 10 12
1, 2, 3	P_1 - X_1	P_2 - X_2	P_3 - X_3
4, 5, 6	P_4 - X_4	P_5 - X_5	P_6 - X_6
7, 8, 9	P_7 - X_7	P_8 - X_8	P_9 - Ø
9, 10	P_{10} - Ø	P_{11} - X_9	P_{12} - X_{10}
11, 12	P_{13} - X_{11}	P_{14} - Ø	P_{15} - X_{12}
13, 14	P_{16} - X_{13}	P_{17} - X_{14}	P_{18} - Ø

Figure 8. Allocation of submatrices for n=14

8. Conclusion

In this article the exchanging interactions in PA of type „allocation of submatrices with identical dimensions in processors of CS with parallel architecture" of the linear equational system, solved through the method of sequential interations are analysed.

The number of transmitted and received words between the threads (branches) of PA is calculated.

Allocations of submatrices are suggested that facilitate the routing algorithms and minimize the number of exchanged words.

to be continued on page 39

As the results show, the **LFU-RBH** outperforms the other algorithms.

Conclusions and Future Work

The conducted experiments show that the introduced improvements result very well in the behavior of the algorithm. Even without the newly proposed MRU section the **LFU-RBH** achieves very good hit ratio, in most cases better than some existing algorithms. Running with the *MRU section* gives a leading position of the **LFU-RBH**, compared with the experimented algorithms.

The use of *data pooling* results in a reduced utilization of the buffer memory. Also, the results for the different reference strings (**RS1** and **RS2**) state that the **LFU-RBH** carries better the simulation of a large area scan situation.

Many other experiments can be conducted to tune the algorithm and to balance between different parameters — hash bits, the length of RB, the number of sections, the length of MRU section and etc.

References

1. Atanassov, I. An Approach for Database Disk Buffering.International Conference of Young Scientists, Plovdiv, Bulgaria, June 2007.
2. Effelsberg, W., T. Haerder.Principles of Database Buffer Management. ACM Digital Library, 1984, ISSN:0362-5915.
3. Goh, C. L., Y. Shu, Z. Huang, B. C. Ooi. Dynamic Buffer Management with Extensible Replacement Policies. ACM Digital Library, 2006, ISSN:1066-8888.
4. Jiang, S., X. Zhang. LIRS: An Efficient Low Interreference Recency Set Replacement Policy to Improve Buffer Cache Performance. ACM Digital Library, 2002, ISBN:1-58113-531-9.
5. Johnson, T., D. Shasha. 2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm. ACM Digital Library, 1994, ISBN:1-55860-153-8.
6. Lee et all. On the Existence of a Spectrum of Policies that Subsumes the Least Recently Used (LRU) and Least Frequently Used (LFU) Policies. ACM Digital Library, 1999, ISSN:0163-5999.
7. O'Neil, E. J., P. E. O'Neill, G. Weikum. The LRU-K Page Replacement Algorithm For Database Disk Buffering. ACM Digital Library, 1993, ISSN:0163-5808.
8. Robinson, J., M. V. Devarakonda. Data Cache Management Using Frequency-based Replacement. ACM Digital Library, 1990, ISSN:0163-5999.
9. Shallahamer, C. A. All About Oracle's Touch Count Data Block Buffer Cache Algorithm. www.orapub.com, 2004.
10. Tanenbaum, A. Modern Operating Systems. Second Edition, Prentice Hall, 2001, ISBN 0-13-092641-8.

Manuscript received on 3.07.2007

Ivaylo Atanasov *graduated from the Technical University — Sofia, Branch Plovdiv, speciality Computer Systems in 1997. In the same university from 2001 he was preparing a PhD, thesis „Algorithms and Data Structures in Client-Server Implementation". Since 2003 he has been working as a full-time assistant in Technical University — Sofia, Branch Plovdiv, Department of Computer Systems and Technologies. Scientific interests: database systems, object-oriented programming, system architectures. He has six publications in area of PhD thesis.*

Contacts:
Department of Computer Systems and Technologies
Technical University-Sofia, Branch Plovdiv
Phone: +359 32 659729,
e-mail: ivo_atan@tu-plovdiv.bg

continuation from 15

The dimension of the submatrix is determined that defines minimal number of exchanged words.

This article „hints" for future directions of work, as:

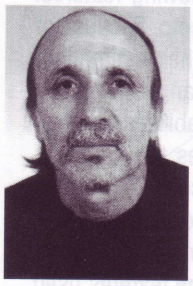
— analysing the other possible allocations of the information in the parallel branches of the given task;

— analysing the exchanging interactions in PA of other tasks in order to find optimal algorithms.

References

1. Seyed H. Roosta. Parallel Processing and Parallel Algorithms: Theory and Computation. Springer, 2000.
2. Cosnard, M., D. Trystram. Parallel Algorithms and Architectures. Thomson Computer Press, 1995.
3. Евреинов,Э. В., Ю. Г. Косарев. Однородные универсальные вычислительные системы высокой производительности. Новосибирск Наука, 1966.
4. Vasilev, N. Main Principles for Searching and Creating Parallel Algorithms, Information Technologies and Control, 2004, 2, ISSN 1312-2622.
5. Wilkinson, B., M. Allen. Parallel Programing. Prentice Hall,1999.

Manuscript received on 12.04.2007



Assoc. Prof., Ph.D Nayden Vasilev *was born in 1943. He graduated the Technical University — Sofia, major Electronics in 1968. His main areas of interest are Parallel Algorithms.*

Contacts:
Technical University of Sofia—branch Plovdiv
Plovdiv 4000,
25 Tzanko Dustabanov Str.
e-mail: mnvasilev@yahoo.com