The Ministry of Education and Science of the Russian Federation

Lobachevsky State University of Nizhni Novgorod

Computing Mathematics and Cybernetics faculty

The competitiveness enhancement program

of the Lobachevsky State University of Nizhni Novgorod

among the world's research and education centers

Strategic initiative

“Achieving leading positions in the field of supercomputer technology

and high-performance computing”

Introduction to MPI

Lecture 12. Estimation of Communication Complexity for Parallel Algorithms

Nizhni Novgorod

2014

Lecture_12_. Estimation of Communication Complexity for Parallel Algorithms

Time delays in case of data transmission via communication channels to ensure interaction of independent processes can to a large extent determine the efficiency of parallel computations. This lecture tackles the issues of analysis of data streams generated in the course of parallel algorithm execution. It gives a general description of data transmission mechanisms, analyzes complexity of basic communication operations and reviews logical representation methods of multiprocessor structure. These issues are studied in more detail by Kumar (1994) and Quinn (2004).

12.1. Overview of Data Transmission Mechanisms

12.1.1. Routing Algorithms

The routing algorithms define the route of data transmission from the sending processor to the processor, which should receive the message. The methods for solving the problem are the following:

– *the optimum ones,* which always define the shortest path for data transmission, and *non-optimum* routing algorithms;

– *deterministic and adaptive methods of choosing routes* (the adaptive algorithms define the route of data transmission depending on the available load of communication channels).

Among the widely used optimum algorithms, there is the class of *dimension-ordered routing methods*. The search for data transmission routes is carried out in the methods for each topology dimension of communication network in turn. Thus, for the two-dimensional grid this approach leads to such type of routing, which involves data transmission first in one direction (for instance, horizontally till the vertical line of processors, which contains the assigned processor, is reached), and then the data is transmitted along the other direction (the given scheme is known as *the XY-routing algorithm*).

12.1.2. Data Transmission Methods

The time necessary for transmitting data between the processors defines the communication overhead of the the duration of parallel algorithm execution in a multiprocessor computer system. The basic set of parameters, which can help to evaluate the data transmission time, consists of the following values:

– **initializing time** (t_s) characterizes the duration of preparing the message for transmission, the search of the route in the network etc.

– **control data transmission time** (t_h) between two neighboring processors (i.e. the processors, connected by a physical data transmission channel); to control data we may refer the message header, the error detection data block etc.;

– **transmission time of one data byte** along a data transmission channel (t_b); the duration of this transmission is defined by the communication channel bandwidth.

There are two main communication methods among the most widely used data transmission techniques (see, for instance, Kumar (1994)). The first one is oriented at transmitting messages as indivisible information blocks (*store-and-forward routing* or *SFR*). In case of this approach the processor, which contains a message for transmission, gets all the amount of data ready for transmission, defines the processor, which should receive the data, and initializes the operation of data transmission. The processor, to which the message has been sent, first receives all the transmitted data and only then begins to send the received message further along the route. The time of data transmission t_{comm} for the method of transmitting the message of m bytes along the route of length l is defined by the expression

$$t_{comm} = t_s + (mt_s + t_h)l .$$

If the messages are long enough, the control data transmission time may be neglected, and the expression for data transmission time may be written in a simplified way:

$$t_{comm} = t_s + mt_h l .$$

The second communication method is based on presenting the transmitted messages as information blocks of smaller sizes (*packets*). Data transmission, as a result, may be reduced to packet communication. In case of this method (*cut-through routing* or *CTR*) the receiving processor may send the data further along the route immediately after receiving the current packet without waiting for the termination of the whole message data transmission. The data transmission time in case of packet communication method will be defined by the following expression:

$$t_{comm} = t_s + mt_b + t_b l .$$

If we compare the obtained expressions, it is possible to notice that in the majority of cases the packet communication leads to faster data transmission. Besides, this approach decreases the need for memory for storing the transmitted data. Different communication channels may be used for packet communication simultaneously. On the other hand, the implementation of the packet communication requires the development of more complex hardware and software. It may also increase the overhead expenses (initialization time and control data transmission time). Deadlocks may also occur in case of packet communication.

12.2. The Complexity Analysis of the Data Transmission Operations

Despite all the variety of the data transmission operations in case of parallel methods of solving complex time-consuming problems, certain procedures of network processor interactions may be referred to the basic communication operations. Such operations are either widely used in parallel computation practice or other message passing operations may be reduced to them. It is also important that there are procedures reverse in their actions to the initial operations among the basic set of communication operations. Thus, for instance, the operation of data transmission from a processor to all the available network processors corresponds to the operation of message accumulation by one processor from all the rest processors. As a result, the consideration of the communication procedures should be done pairwise. It is useful as the execution algorithm of the direct and the reverse operations may be in many cases obtained proceeding from the general statements.

Such network topologies as a ring and two-dimensional grid will be further used as examples for the consideration of the basic data transmission operations. For the two-dimensional grid it is also assumed that there are data transmission channels between the bordering processors in the rows and columns of the grid (i.e. the network topology is a torus).

12.2.1. Data Transmission between Two Network Processors

The complexity of this communication operation may be obtained by means of substitution of the maximum path (the network diameter – see Table 12.1) into the expression for data transmission time in case of various communication methods.

Table 12.1. Data transmission time between two processors

Topology	**Data Transmission**	**Packet Communication**
Ring	$t_н + mt_к \lfloor p/2 \rfloor$	$t_н + mt_к + t_c \lfloor p/2 \rfloor$
Grid-torus	$t_н + 2mt_к \lfloor \sqrt{p}/2 \rfloor$	$t_н + mt_к + 2t_c \lfloor \sqrt{p}/2 \rfloor$
Hypercube	$t_н + mt_к \log_2 p$	$t_н + mt_к + t_c \log_2 p$

12.2.1. Data Transmission from One Processor to All the Network Processors

One-to-all broadcast or *single-node broadcast* (of the same message) is one of the most widely used communication operations. *Single-node accumulation* consists in receiving the messages by one of the processors from all the other network processors. Such operations are used in

particular in matrix-vector multiplication implementation, in solving linear equation systems by the Gauss method, as well as in searching for the shortest paths etc.

The simplest way to realize broadcast operation is to carry it out as a sequence of pairwise network processor interactions. However, in case of this approach the greater part of broadcasting is excessive. So a more efficient communication algorithm may be used. First we will describe the message transmission method, and then we will consider packet communication.

Message transmission. In case of the **ring** topology the sending processor may initiate data transmission to two neighbors at once. These processors in their turn send the message further in the ring. The complexity of the operation execution in this case will be defined by the following relation:

$$t_{comm} = (t_s + mt_b)\lceil p/2 \rceil .$$

For the **grid-torus** topology the broadcasting algorithm may be obtained basing on the data transmission method, which was used for the ring topology. Thus, broadcasting may be carried out as a two-stage procedure. At the first stage we arrange data broadcasting to all the processors of the network, which are located on the same horizontal line of the grid as the sending processor. During the second stage the processors, which have received the data copy at the first stage, send the messages along the corresponding vertical lines. The estimation of broadcasting duration in accordance to the described algorithm, is defined by the following formula:

$$t_{comm} = 2(t_s + mt_b)\lceil \sqrt{p}/2 \rceil$$

For the **hypercube** broadcasting may be carried out as an *N*-stage data transmission procedure. During the first stage the sending processor sends data to one of the neighbors (for instance, along the first dimension). As a result, there are two processors, which have the copies of the data after the first stage (these results may be also interpreted as bisecting of the initial hypercube and obtaining two identical in size hypercubes of *N-1* dimensionality, each of them has a copy of the initial message). At the second stage the two processors engaged at the first stage send messages to their neighbors along the second dimension etc. As a result of this broadcasting the operation execution time is estimated by the following expression:

$$t_{comm} = (t_s + mt_b)\log_2 p .$$

The comparison of the obtained expressions for broadcasting execution duration demonstrates that the hypercube topology shows the best results. Moreover, it is possible to demonstrate that the given result is optimum for the selected communication method based on message transmission.

Packet communication. The broadcast algorithm for the **ring** topology may be obtained by means of logical presentation of the ring structure as a hypercube. As a result, the sending processor sends the data to the processor, which is at *p/2* distance from the initial processor during the first broadcast stage. Further, during the second stage the two processors, which already have the data after the first stage, transmit the data to the processors, which are located at *p/4* distance etc. The time complexity of the broadcast in case of this method is defined by the following expression:

$$t_{comm} = \sum_{i=1}^{\log_2 p} (t_s + mt_b + t_h p / 2^i) = (t_s + mt_b)\log_2 p + t_h(p-1)$$

(as previously, if messages are big enough, control data transmission time may be neglected).

For the **grid-torus** topology the broadcast algorithm may be obtained using the method of data transmission applied to the ring network structure. The same way of generalization, that is used for message transmission method, may be also applied. The algorithm, which is obtained, is characterized by the following statement for estimating execution time:

$$t_{comm} = (t_s + mt_b)\log_2 p + 2t_h(\sqrt{p} - 1).$$

The packet communication broadcast algorithm for the **hypercube** (and correspondingly, execution time estimations) does not differ from the variant for message transmission method.

12.3. Estimation of Communication Complexity for Clusters

One of the most efficient methods of organizing the communication network for cluster computer systems is using hubs and switches. In these cases the cluster network topology is the complete graph. There are, however, certain limitations on communication operation simultaneity in the cluster. Thus, data transmission at any given moment of time may be executed only between two processors, if hubs are used. Switches may provide the interactions of several non-intersecting pairs of processors.

Another solution, which is widely used in creating clusters, consists in using packet communication method (which is realized, as a rule, on the basis of TCP/IP protocol). This method is used as the basic means of executing communication operations.

1. If we choose for the further analysis the clusters of this widely used type (the complete graph topology, packet communication method), then the time complexity of the communication

operation between two processors may be estimated according to the following formula (*model A*):

$$t_{comm}(m) = t_s + m * t_b + t_h ,$$

the estimation of this type is caused by the expression of packet communication method, when the path length of data transmission is $l = 1$. Such an approach is quite possible. However, it is possible to notice that in this model the time of data preparation t_s is assumed to be constant (it does not depend on the amount of the transmitted data). The time of control data transmission t_h does not depend on the number of the transmitted packets etc. These assumptions do not fully coincide with the real situation, and the time estimations obtained with the help of this model may be not accurate enough.

2. If we take into account all these considerations, we may specify a new enhanced model which provide the estimation of the time transmission complexity for data transmission between two processors in accordance with the following expression (*model B*):

$$t_{comm} = \begin{cases} t_{s_0} + m \cdot t_{s_1} + (m + V_h) \cdot t_b , & n = 1 \\ t_{s_0} + (V_{\max} - V_h) \cdot t_{s_1} + (m + V_h \cdot n) \cdot t_b , & n > 1 \end{cases},$$

where $n = \lceil m / (V_{\max} - V_h) \rceil$ is the number of packets, into which the transmitted message is partitioned, value $V_{\max}$ defines the maximum size of the packet, which may be delivered in the network, and V_h is the volume of control data in each of the transmitted packets. It should be noted that the constant t_{s_0} in the above given relations characterizes the hardware latency component and depends on the parameters of the network equipment being used; the value t_{s_1} defines the time for preparing a data byte for transmission. As a result, the latency value

$$t_s = t_{s_0} + \nu \cdot t_{s_1}$$

increases linearly depending on the amount of the transmitted data. It is assumed that data preparation for transmitting the second and all the further packets may be combined with transmitting the previous packets. Thus, latency cannot exceed the following value

$$t_s = t_{s_0} + (V_{\max} - V_h) \cdot t_{s_1} .$$

Besides latency it is possible to specify the rule for computation of data transmission time in the suggested expressions for estimating the time communication complexity:

$$(m + V_h \cdot n) \cdot t_b .$$

It makes possible to take into account the effect of the increase of the transmitted data amount, if the number of the transmitted packets is increasing due to addition of the control information (packet headers).

3. At the end of the problem analysis it should be noted that it is necessary to estimate the values of the parameters for the relations being used in order to use the above described models in practice. In this respect it may be useful to use simpler methods of computing the time expenses for data transmission. One of the best known schemes of this type is the approach (the Hockney model; see, for instance, Hockney (1994)), which estimates the complexity of the communication between two processors according to the expression (*model C*):

$$t_{comm}(m) = t_s + m\,t_b\,.$$

4. In order to check whether these models are adequate to real data transmission processes, we will show the results of the computational experiments carried out in the network of the multiprocessor cluster of Nizhny Novgorod State University (computers IBM PC Pentium 4 1300 Mhz and Fast Ethernet). Communication operations were realized with the help of MPI library in these experiments.

A number of the experiments were carried out for estimating the model parameters:

- the latency value t_s for the models A and C was defined as the time of transmitting the message of zero length;
- the value of bandwidth R was set as the maximal transmission speed observed in the course of experiments, i.e.:

$$R = \max_m \quad (t_{comm}(m)/m)\,,$$

and it was assumed that $t_b = 1/R$;

- the values t_{s_0} and t_{s_1} were estimated by means of linear approximation of the data transmission time for messages beginning with 0 size up to $v_{\max}$ size,
- there were also set $v_{\max}$ =1500 bytes and v_h =78 bytes.

The data was transmitted between two cluster processors in the experiments. The size of the transmitted messages varied from 0 to 8 Mb. Each operation was carried out repeatedly (more than 100000) in order to obtain more accurate estimations. After that the results were averaged. For illustration the results of an experiment are given below when the size of the transmitted messages varied from 2000 up to 60000 bytes.

The numeric data concerning the errors of the above described models of communication complexity are given in Table 12.2 (the value of the error is given as relative deviation from the real time of data transmission operation execution).

Table 12.2. The errors of the complexity models for data transmission (according to the results of the computational experiments)

Message	Transmission time (msec)	The error of the theoretical estimation of data transmission time (%)

size (byte)		Model A	Model B	Model C
2000	495	33.45%	7.93%	34.80%
10000	1184	13.91%	1.70%	14.48%
20000	2055	8.44%	0.44%	8.77%
30000	2874	4.53%	-1.87%	4.76%
40000	3758	4.04%	-1.38%	4.22%
50000	4749	5.91%	1.21%	6.05%
60000	5730	6.97%	2.73%	7.09%

The results of the experiments demonstrate that the estimations of data transmission complexity according to model B have the least error.

It should be also noted that model C accuracy may appear to be sufficient for the preliminary analysis of the time expenses on communication operations. Besides, this model is the simplest one among all the models, which have been discussed here. With regard to the latter fact we will be using model C (the Hockney model) in all the further sections for estimating the data transmission complexity. Moreover, further we will use the following format for the Hockney model (see Hockney (1994)):

$$t_{comm}(m) = \alpha + m/\beta ,$$

where α is the latency of the data transmission network (i.e. $\alpha = t_s$), β is the network bandwidth (i.e. $\beta = R = 1/t_b$).

12.4. Examples of practical task solution time model construction

Lectures 11 and 12 offer methods of solution time modeling for the developed algorithms. Let us try to put the methods into practice and solve the matrix-vector multiplication and matrix multiplication problems considered earlier.

12.4.1. The problem of matrix-vector multiplication

See the problem definition and program implementation of parallel methods in Lectures 6 and 7. We will only list the result of efficiency analysis for parallel implementations.

12.4.1.1. Rowwise data decomposition

To analyze the efficiency of parallel computations, two kinds of estimations will be formed henceforward. To form the first type of them algorithm complexity is measured by the number of computational operations that are necessary for solving the given problem (without taking into

account the overhead caused by data communications among the processors); the duration of all elementary computational operations (for instance, addition and multiplication) is considered to be the same. Besides, the obtained constants are not taken into consideration in relations. It provides to obtain the order of algorithm complexity and, as a result, in most cases such estimations are rather simple and they can be used for the initial efficiency analysis of the developed parallel algorithms and methods.

The second type of estimation is aimed at forming as many exact relationships for predicting the execution time of algorithms as possible. Such estimations are usually obtained with the help of refinement of the expressions resulting from the first stage. For that purpose the parameters, which determine the execution time, are introduced in the existing relations; time complexity of communication operations are estimated; all the necessary constants are stated. The accuracy of the obtained expressions is examined with the help of computational experiments. On the basis of their results the time of executed computations is compared to the theoretically predicted estimation of the execution time. As a result, such estimations are, as a rule, more complex, but they make it possible to estimate the efficiency of the developed parallel computation methods more precisely.

Let us consider the time complexity of the algorithm of matrix-vector multiplication. If matrix A is square ($m=n$), the sequential algorithm of matix-vector multiplication has the complexity $T_1=n^2$. In case of parallel computations each processor performs multiplication of only a part (stripe) of the matrix A by the vector b. The size of these stripes is equal to n/p rows. In case of computing the inner product of one matrix row by a vector, it is necessary to perform the n multiplications and $(n\text{-}l)$ additions. Therefore, computational complexity of the parallel algorithm is determined as:

$$T_p=n^2/p. \qquad (12.1)$$

Taking into account this estimation, the criteria of speedup and efficiency of the parallel algorithm are:

$$S_p = \frac{n^2}{n^2/p} = p\,, \qquad E_p = \frac{n^2}{p\cdot(n^2/p)} = 1\,. \qquad (12.2)$$

The estimations (12.1), (12.2) of the computation execution time are expressed in the number of operations. Besides, they are formed without taking into consideration the execution of data communication operations. Let us use the above mentioned assumptions that the executed multiplications and additions are of equal duration τ. Besides, let us assume that the computer system

is homogeneous, i.e. all the processors of the system have the same performance. With regard to the introduced assumptions, the computation time of the parallel algorithm is:

$$T_p(calc) = \lceil n/p \rceil \cdot (2n-1) \cdot \tau$$

($\lceil\ \rceil$ is denoted rounding up to the nearest integer number).

As it has been mentioned before, this operation can be executed in $\lceil \log_2 p \rceil$ iterations[1.)]. At the first iteration the interacting pairs of processors exchange messages of size $w\lceil n/p \rceil$ (w is the size of one element of the vector c in bytes). At the second iteration the size becomes doubled and is equal to $2w\lceil n/p \rceil$ etc. As a result, the all gather operation execution time when the Hockney model is used can be represented as:

$$T_p(comm) = \sum_{i=1}^{\lceil \log_2 p \rceil} (\alpha + 2^{i-1} w \lceil n/p \rceil / \beta) = \alpha \lceil \log_2 p \rceil + w \lceil n/p \rceil (2^{\lceil \log_2 p \rceil} - 1)/\beta ,$$

where α is the latency of data communication network, β is the network bandwidth. Thus, the total time of parallel algorithm execution is

$$T_p = (n/p) \cdot (2n-1) \cdot \tau + \alpha \cdot \log_2 p + w(n/p)(p-1)/\beta \quad . \qquad (12.3)$$

(to simplify the expression , it was assumed that the values n/p and $\log_2 p$ are whole numbers).

Let us analyze the results of the computational experiments carried out in order to estimate the efficiency of the discussed parallel algorithm of matrix-vector multiplication. Besides, the obtained results will be used for the comparison of the theoretical estimations and experimental values of the computation time. Thus, the accuracy of the obtained analytical relations will be checked. The experiments were carried out on the computational cluster on the basis of the processors Intel XEON 4 EM64T, 3000 Mhz and the network Gigabit Ethernet under OS Microsoft Windows Server 2003 Standard x64 Edition.

Now let us describe the way the parameters of the theoretical dependencies (values τ, w, α, β) were evaluated. To estimate the duration τ of the basic scalar computational operation, we solved the problem of matrix-vector multiplication using the sequential algorithm. The computation time obtained by this method was divided into the total number of the operations performed. As a result of the experiments the value of τ was equal to 1.93 nsec. The experiments carried out in order to determine the data communication network parameters demonstrated the value of laten-

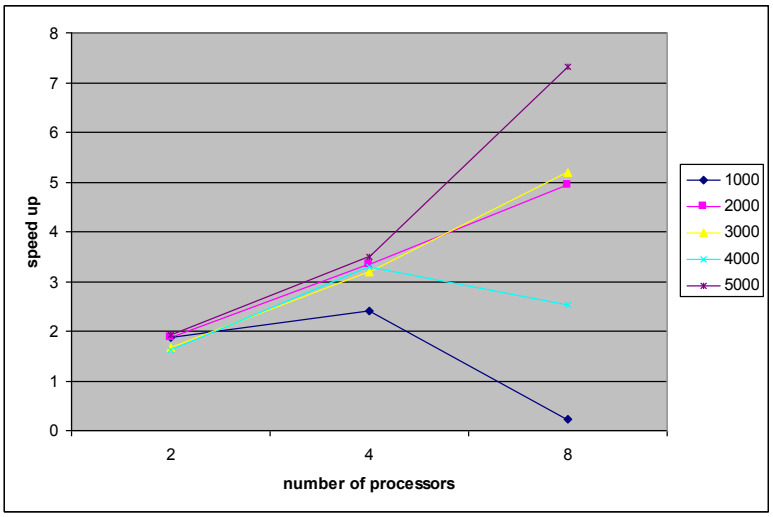
[1)] Let us assume that the topology of the computer system allows to carry out this efficient method of all gather operation (it is possible, in particular, if the structure of the data communication network is a hypercube or a complete graph).

cy α and bandwidth β correspondingly 47 msec and 53.29 Mbyte/sec. All the computations were performed over the numerical values of the double type, i.e. the value *w* is equal to 8 bytes.

The results of the computational experiments are shown in Table 12.3. The experiments were carried out with the use of 2, 4 and 8 processors. The algorithm execution time is given in seconds.

Table 12.3 The results of the computational experiments for the parallel algorithm of matrix-vector multiplication with rowwise block-striped data decomposition

Matrix Size	Sequential Algorithm	Parallel Algorithm					
		2 processors		4 processors		8 processors	
		Time	Speed Up	Time	Speed Up	Time	Speed Up
1000	0,0041	0,0021	1,8798	0,0017	2,4089	0,0175	0,2333
2000	0,016	0,0084	1,8843	0,0047	3,3388	0,0032	4,9443
3000	0,031	0,0185	1,6700	0,0097	3,1778	0,0059	5,1952
4000	0,062	0,0381	1,6263	0,0188	3,2838	0,0244	2,5329
5000	0,11	0,0574	1,9156	0,0314	3,4993	0,0150	7,3216



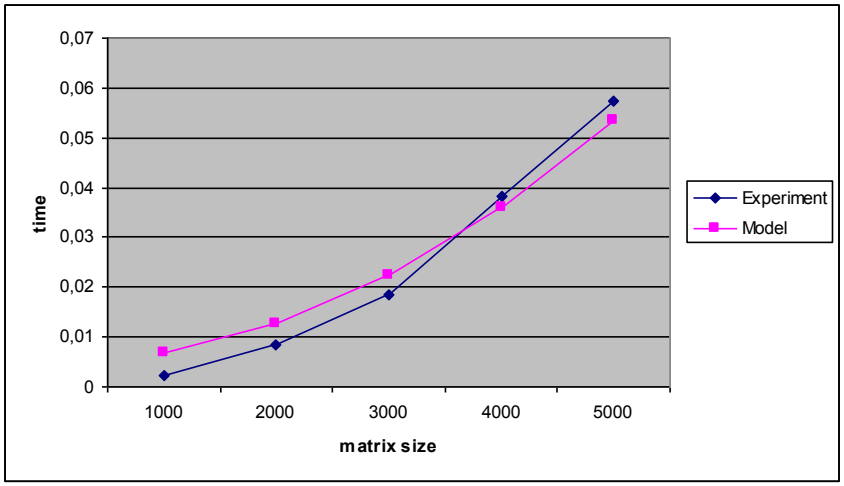
Figures 12.1 Speedup for parallel matrix-vector multiplication (rowwise block-striped matrix decomposition)

The comparison of the experiment execution time T_p^* and the theoretical time T_p calculated in accordance with the expression (12.3) is shown in Table 12.4. It is also shown graphically in Figures 12.1 and 12.2.

Table 12.4 The comparison of the experimental and theoretical execution time for parallel algorithm of matrix-vector multiplication based on rowwise matrix decomposition

Matrix Size	2 processors	4 processors	6 processors

	T_p (model)	T_p^*	T_p (model)	T_p^*	T_p (model)	T_p^*
1000	0,0069	0,0021	0,0108	0,0017	0,0152	0,0175
2000	0,0132	0,0084	0,0140	0,0047	0,0169	0,0032
3000	0,0235	0,0185	0,0193	0,0097	0,0196	0,0059
4000	0,0379	0,0381	0,0265	0,0188	0,0233	0,0244
5000	0,0565	0,0574	0,0359	0,0314	0,0280	0,0150



Figures 12.2 Experimental and theoretical execution time with respect to matrix size (rowwise block-striped matrix decomposition, 2 processors)

12.4.2. Matrix multiplication

See Lectures 8-9 for problem description and parallel method implementation. Here we will indicate only the results of parallel implementation efficiency analysis.

12.4.2.1. Block-striped data decomposition scheme

Let us estimate the efficiency of the first matrix multiplication parallel algorithm.

The total time complexity of the sequential algorithm, as it has been stated earlier, is proportional to n^3. In case of the parallel algorithm each processor multiplies the stripes of the matrix A and the matrix B at each iteration (the stripe size is equal to n/p and, as a result, the total number of the multiplication operations performed is equal to n^3/p^2). As the number of the algorithm iterations is the same as the number of processors the complexity of the parallel algorithm, with no account for data communication, may be evaluated by means of the following expression:

$$T_p = (n^3 / p^2) \cdot p = n^3 / p \,. \qquad (12.4)$$

With regard to this estimation, the speedup and efficiency of the given parallel algorithm of matrix multiplication look as follows:

$$S_p = \frac{n^3}{(n^3/p)} = p \quad \text{and} \quad E_p = \frac{n^3}{p \cdot (n^3/p)} = 1. \tag{12.5}$$

Thus, the general efficiency analysis gives ideal characteristics of the parallel computation efficiency. To specify the obtained relations we should estimate more precisely the number of computational operations of the algorithm and take into account the overhead of data communications among the processors.

With regards to the number and the duration of the operations the time for carrying out the computations for parallel algorithm may be estimated as follows:

$$T_p(calc) = (n^2 / p) \cdot (2n - 1) \cdot \tau \tag{12.6}$$

(where, as previously, τ is the execution time of an basic computational operation).

For the purpose of estimating the communication complexity of parallel computations we will assume that all data communication operations among the processors in the course of an algorithm iteration may be executed in parallel. The amount of the data transmitted among the processors is determined by the stripe size and is equal to *n/p* rows or columns of size *n*. The total number of parallel data communication operations is equal to the number of algorithm iterations minus one (at the last iteration data communication is not compulsory). Thus, the time complexity estimation for the data communication operations performed may be evaluated as:

$$T_p(comm) = (p - 1) \cdot (\alpha + w \cdot n \cdot (n/p)/\beta), \tag{12.7}$$

where α is the latency, β is the network bandwidth, and *w* is the size of the matrix element in bytes.

With regard to the relations obtained the total execution time for the parallel algorithm of matrix multiplication can be estimated by the following expression:

$$T_p = (n^2 / p)(2n - 1) \cdot \tau + (p - 1) \cdot (\alpha + w \cdot n \cdot (n/p)/\beta). \tag{12.8}$$

The results of the computational experiments are shown in Table 12.5. The experiments were performed with the use of 2, 4 and 8 processors.

Table 12.5. The results of the computational experiments for the first parallel algorithm of matrix multiplication based on the block-striped data decomposition

Matrix Size	Serial Algorithm	2 processors		4 processors		8 processors	
		Time	Speed Up	Time	Speed Up	Time	Speed Up

500	0,8752	0,3758	2,3287	0,1535	5,6982	0,0968	9,0371
1000	12,8787	5,4427	2,3662	2,2628	5,6912	0,6998	18,4014
1500	43,4731	20,9503	2,0750	11,0804	3,9234	5,1766	8,3978
2000	103,0561	45,7436	2,2529	21,6001	4,7710	9,4127	10,9485
2500	201,2915	99,5097	2,0228	56,9203	3,5363	18,3303	10,9813
3000	347,8434	171,9232	2,0232	111,9642	3,1067	45,5482	7,6368

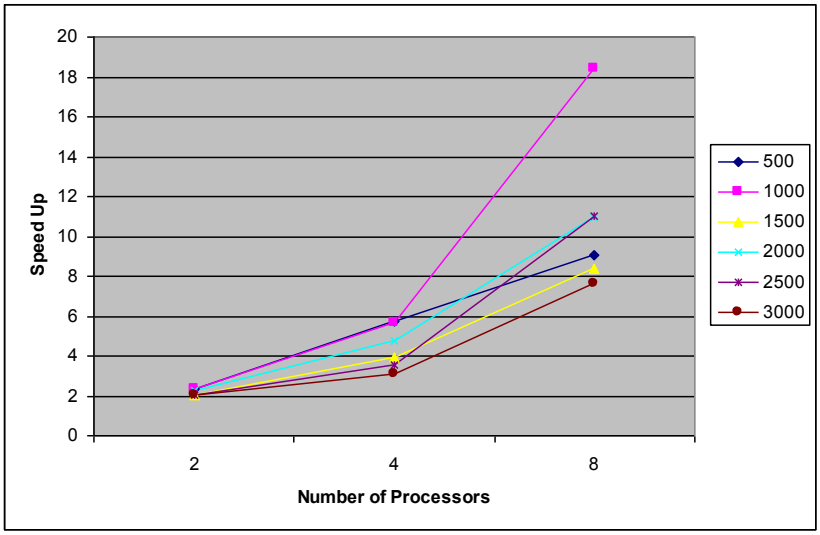


Figure 12.3 Speedup for the first parallel algorithm of matrix multiplication (block-striped matrix decomposition)

The comparison of the experimental execution time T_p^* and the theoretical time T_p from expression (12.8) is given in Table 12.6 and in Figure 12.4.

Table 12.6. The comparison of the experimental and theoretical execution time of the first matrix multiplication parallel algorithm based on the block-striped data decomposition

Matrix Size	2 processors		4 processors		8 processors	
	T_p	T_p^*	T_p	T_p^*	T_p	T_p^*
500	0,8243	0,3758	0,4313	0,1535	0,2353	0,0968
1000	6,51822	5,4427	3,3349	2,2628	1,7436	0,6998
1500	21,9137	20,9503	11,1270	11,0804	5,7340	5,1766
2000	51,8429	45,7436	26,2236	21,6001	13,4144	9,4127
2500	101,1377	99,5097	51,0408	56,9203	25,9928	18,3303
3000	174,6301	171,9232	87,9946	111,9642	44,6772	45,5482

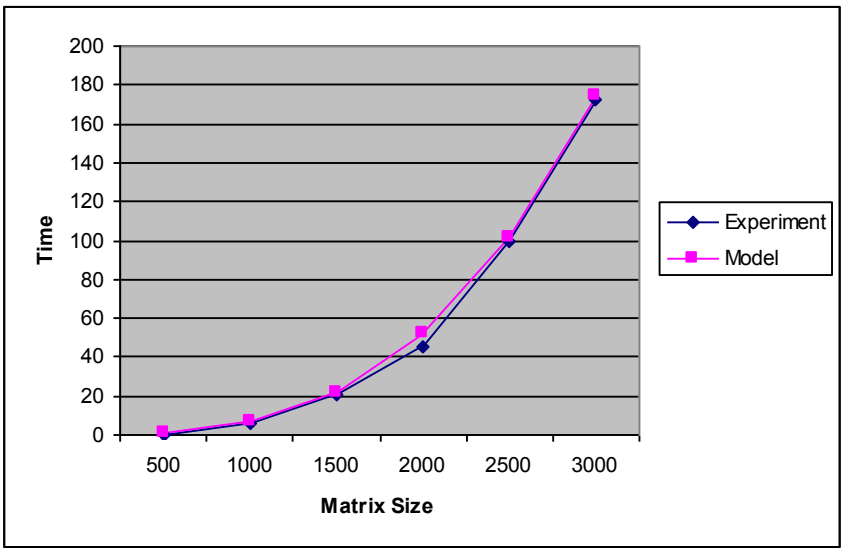


Figure 12.4 Theoretical and experimental execution time with respect to matrix size (block-striped matrix decomposition, 2 processors)

12.4.2.2. Checkerboard block matrix data partitioning (Fox algorithm)

Let us evaluate the computational complexity of the Fox algorithm. To formulate the required estimations we will suppose that all the previous assumptions are met, i.e. all matrices are square, their sizes are $n \times n$, the block grid is square and its size is equal to q (i.e. the size of all blocks is $k \times k$, $k=n/q$), processors form a square grid and their number is $p=q^2$.

As it has been previously mentioned, the execution of the Fox algorithm requires q iterations, during which each processor multiplies its current blocks of the matrices A and B, and adds the multiplication results to the current block of the matrix C. With regard to the above mentioned assumptions, the total number of the executed operations will be of the order n^3/p. As a result, the speedup and efficiency of the algorithm look as follows:

$$S_p = \frac{n^3}{(n^3/p)} = p \quad \text{and} \quad E_p = \frac{n^3}{p \cdot (n^3/p)} = 1 . \tag{12.9}$$

The general efficiency analysis indicates that parallel computation efficiency is ideal. To make the obtained relations more precise we will estimate more exactly the number of algorithm computational operations and take into account the overhead related with data communications among the processors.

Let us evaluate the number of computational operations. The complexity of scalar multiplication of the block row of the matrix A by the block column of the matrix B may be estimated as $2(n/q)-1$. The number of rows and columns in the blocks is equal to n/q. As a result, the time

complexity of block multiplication appears to be equal to $(n^2/p)(2n/q-1)$. The addition of the blocks requires n^2/p operations. With regard to the above given expressions the computational time of the Fox algorithm may be estimated in the following way:

$$T_p(calc) = q[(n^2/p)\cdot(2n/q-1)+(n^2/p)]\cdot\tau\ . \qquad (12.10)$$

(as previously that τ is the execution time of an basic computational operation).

Let us estimate now the overhead on data communications among the processors. One of the processors of the processor grid row transmits its matrix *A* block to the rest of the grid row processors at each iteration. As it has been mentioned previously, the execution of this operation may be provided in $log_2 q$ steps, if the topology of the network is a hypercube or a complete graph. As a result, the time complexity of data communications in accordance with the Hockney model may be estimated as follows:

$$T_p^1(comm) = \log_2 q\,(\alpha + w(n^2/p)/\beta) \qquad (12.11)$$

where α is the latency, β is the network bandwidth, *w* is the size of a matrix elements in bytes. In case of the ring topology the expression for time estimation looks as follows:

$$\tilde{T}_p^1(comm) = (q/2)(\alpha + w(n^2/p)/\beta)\ .$$

After multiplying the matrix blocks the processors send their matrix *B* blocks to the processors, which are upper neighbors in the processor grid columns (the first row processors send their data to the last row processors). These operations may be carried out by the processors in parallel and, thus, the time complexity of these communication operations is the following:

$$T_p^2(comm) = \alpha + w\cdot(n^2/p)/\beta\ . \qquad (12.12)$$

After the summation of all the obtained expressions, it becomes clear that the total execution time for the Fox algorithm may be defined by means of the following relations:

$$\begin{aligned} T_p &= q[(n^2/p)\cdot(2n/q-1)+(n^2/p)]\cdot\tau + q\log_2 q\,(\alpha + w(n^2/p)/\beta) + (q-1)\cdot(\alpha + w(n^2/p)/\beta) = \\ &= q[(n^2/p)\cdot(2n/q-1)+(n^2/p)]\cdot\tau + (q\log_2 q + (q-1))(\alpha + w(n^2/p)/\beta) \end{aligned}$$

(12.13)

(it should be reminded that parameter q defines the size of the processor grid and $q = \sqrt{p}$).

The results of the experiments with the use of 4 and 9 processors are given in Table 12.7.

Table 12.7 The Results of the computational experiments for estimating the Fox parallel algorithm efficiency

Matrix	Serial Algorithm	Parallel Algorithm

Size		4 processors		9 processors	
		Time	Speed Up	Time	Speed Up
500	0,8527	0,2190	3,8925	0,1468	5,8079
1000	12,8787	3,0910	4,1664	2,1565	5,9719
1500	43,4731	10,8678	4,0001	7,2502	5,9960
2000	103,0561	24,1421	4,2687	21,4157	4,8121
2500	201,2915	51,4735	3,9105	41,2159	4,8838
3000	347,8434	87,0538	3,9957	58,2022	5,9764

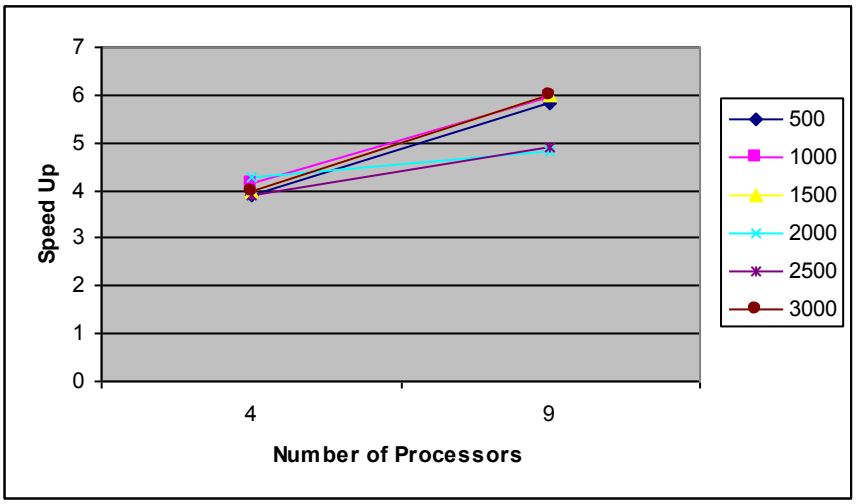


Figure 12.5 Speedup of the Fox Parallel Algorithm with Respect to Number of Processors

The comparison of the experiment execution time T_p^* and the theoretical time T_p, calculated according to expression 12.13, is shown in Table 12.8 and in Figure 12.6.

Table 12.8. The comparison of the experimental and theoretical execution time for the Fox parallel algorithm

Matrix Size	4 processors		9 processors	
	T_p	T_p^*	T_p	T_p^*
500	0,4217	0,2190	0,2200	0,1468
1000	3,2970	3,0910	1,5924	2,1565
1500	11,0419	10,8678	5,1920	7,2502
2000	26,0726	24,1421	12,0927	21,4157
2500	50,8049	51,4735	23,3682	41,2159
3000	87,6548	87,0538	40,0923	58,2022

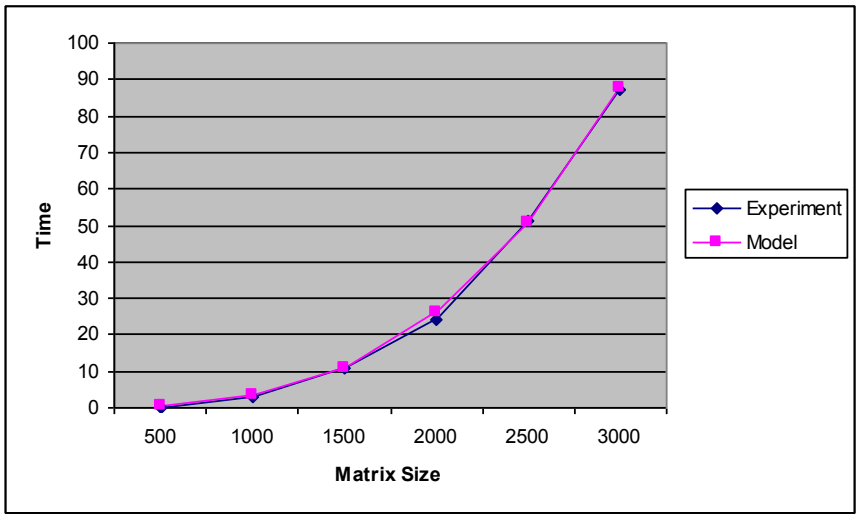


Figure 12.8 Experimental and theoretical execution time of the Fox parallel algorithm with respect to matrix size (checkerboard block matrix decomposition, 4 processors)

12.5. References

The works by Kumar (1994) and Quinn (2004) may be recommended as additional teaching material on the problems discussed in the subsection.

Creating models for estimating the time of communication operations is widely discussed in many papers. Such works as Culler, et al. (1996), Skillicorn and Talia (1998), Andrews (2000) might be of use. The Hockney model was first published in Hockney (1994). Model B from subsection 3.4 is presented in Gergel, Strongin (2001).

Discussions

1. What basic characteristics are used for the estimation of the data transmission network topology? Give the values of the characteristics for the following types of communication structures (a complete graph, a linear array, a grid etc.)
2. What basic methods are applied to routing the data transmitted in the network?
3. What are the basic methods of data transmission? Give the analytical estimations of the execution time for these methods.
4. What data transmission operations may be selected as the basic ones?
5. What are the execution algorithms of one-to-all broadcast for the ring, the grid and the hypercube topologies? Give the estimations of the time complexity for these algorithms.
6. What are the execution algorithms of all-to-all broadcast for the ring, the grid and the hypercube topologies? Give the estimations of the time complexity for these algorithms.
7. What are the possible execution algorithms of reduction? Which of them is the best as far as the execution time is concerned?
8. What does the execution algorithm of the circular shift consist in?
9. Why is it efficient to use logical topologies? Give the examples of the algorithms for logical presentation of communication network structure.

10. How do the models for estimating the execution time of data transmission in cluster computer systems differ from one another? Which model is the most accurate? Which of them may be used for the preliminary analysis of the time complexity of the communication operations?

12.6. Practive

1. Develop the execution algorithms of the basic data transmission operations for the network topology in the form of a three-dimensional grid.
2. Develop the execution algorithm of the basic data transmission operations for the network topology in the form of a binary tree.

References

Gergel, V.P., Strongin, R.G. (2001, 2003 - 2 edn.). Introduction to Parallel Computations. - N.Novgorod: University of Nizhni Novgorod (In Russian)

Culler, D.E., et al. (1996). LogP: A practical model for parallel computation. – Comm. Of the ACM, 39, 11, pp. 75-85.

Hockney, R. (1994). The communication challenge for MPP: Intel Paragon and Meiko CS-2. – Parallel Computing, 20 (3), pp. 389-398.

Kumar V., Grama, A., Gupta, A., Karypis, G. (1994). Introduction to Parallel Computing. - The Benjamin/Cummings Publishing Company, Inc. (2nd edn., 2003)

Quinn, M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.

Skillicorn, D.B., Talia, D. (1998). Models and languages for parallel computation. – ACM Computing surveys, 30, 2.