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 PARALLEL PROGRAMMING

Lectures 13 and 14. Parallel Methods for Matrix Multiplication

for Systems with Distributed Memory

Nizhni Novgorod

2014

Lectures_13,14_. Parallel Methods for Matrix Multiplication for Systems with Distributed Memory

Matrix multiplication is one of the essential problems in matrix calculations. This Lection discusses several parallel algorithms for carrying out the operation. Two of them are based on block-striped data decomposition scheme. The other two methods are based on checkerboard block scheme decomposition. They are the well known the Fox algorithm and the Cannon method.

LECTURE 13

Multiplying an $m \times n$ matrix *A* with *m* rows and *n* columns and an $n \times l$ matrix *B* with *n* rows and *l* columns produces an $m \times l$ matrix *C* with *m* rows and *l* columns. Each element of the matrix *C* is calculated according to the formula:

$$c_{ij} = \sum_{k=0}^{n-1} a_{ik} \cdot b_{kj}, 0 \le i < m, 0 \le j < l . \quad (13.1)$$

As it can be seen in (13.1), each element of the matrix *C* is the result of the inner product of the corresponding row of the matrix *A* and column of the matrix *B*:

$$c_{ij} = (a_i, b_j^T), a_i = (a_{i0}, a_{i1}, ..., a_{in-1}), b_j^T = (b_{0j}, b_{1j}, ..., b_{n-1j})^T . \quad (13.2)$$

This algorithm executes $m \cdot n \cdot l$ multiplications and the same number of additions of the initial matrix elements. In case of square matrices, the size of which is $n \times n$, the number of the executed operations is the order $O(n^3)$. There are also sequential matrix multiplication algorithms of smaller computational complexity (for instance, the Strassen algorithm). But studying these algorithms though requires certain efforts and for simplicity we will use the above described sequential algorithm as the basis for parallel method development in this section. We will also assume further that all matrices are square and their sizes are $n \times n$.

The sequential matrix multiplication algorithm includes three nested loops:

```
// Algorithms 13.1
// Sequential matrix multiplication algorithm
double MatrixA[Size][Size];
double MatrixB[Size][Size];
double MatrixC[Size][Size];
int i,j,k;
...
for (i=0; i<Size; i++){
  for (j=0; j<Size; j++){
    MatrixC[i][j] = 0;
    for (k=0; k<Size; k++){
      MatrixC[i][j] = MatrixC[i][j] + MatrixA[i][k]*MatrixB[k][j];
    }
  }
```

```
}
```

This algorithm is an iterative procedure and calculates sequentially the rows of the matrix *C*. In fact, a result matrix row is computed per outer loop (loop variable *i*) iteration (see Figure 13.1)

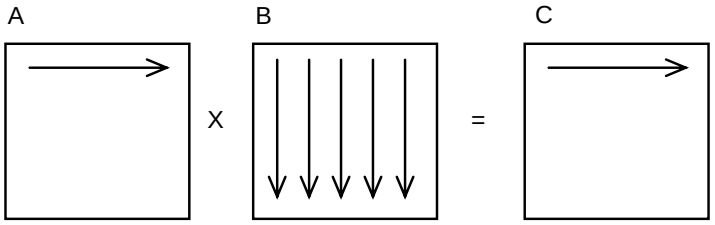

Figure 13.1. During the first iteration of loop variable *i* the first matrix *A* row and all the columns of matrix *B* are used to compute the elements of the first result matrix *C* row

The lecture describes two parallel matrix multiplication algorithms where the matrixes *A* and *B* are decomposed into continuous sequences of rows or columns (*stripes*).

1. The first algorithm. The algorithm is an iterative procedure, the number of iterations is equal to the number of subtasks. Each subtask holds a row of the matrix *A* and a column of the matrix *B* at each algorithm iteration. At each iteration the scalar products of rows and columns containing in the subtasks are computed, and the corresponding elements of the result matrix *C* are obtained. After completing of all iteration computations the columns of matrix *B* must be transmitted so that subtasks should have new columns of the matrix *B* and new elements of the matrix *C* could be calculated. This transmission of columns among the subtasks must be executed in such a way that all the columns of matrix *B* should have appeared in each subtask sequentially.

A possible simple scheme to provide the required communications of the columns of matrix *B* among the subtasks is to present the topology of the information dependencies of the subtasks as a ring structure. In this case the subtask i, $0 \le i < n$, will transmit its column of matrix *B* to the subtask $i+1$ at each iteration (in accordance with the ring structure subtask $n-1$ transmits its data to the subtask 0) – see Figure 13.2. After the algorithm termination the required condition will be provided, i.e. all the columns of matrix *B* will appear sequentially in each subtask.

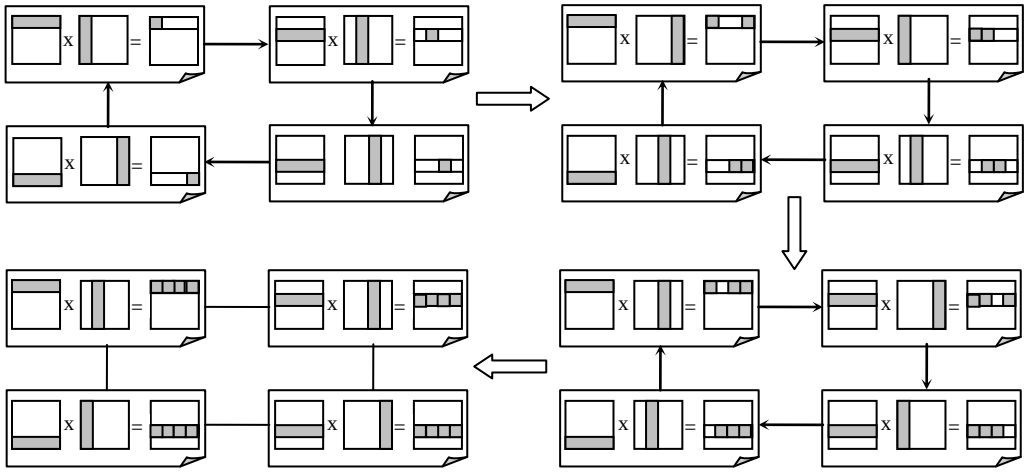

Figure 13.2. General scheme of data communications for the first parallel algorithm of matrix multiplication in case of block-striped decomposition

Figure 13.2 presents the iterations of the matrix multiplication algorithm for the case when matrices have four rows and four columns (n=4). At the beginning of the computations each subtask i, $0 \le i < n$, holds i-th row of the matrix A and i-th column of the matrix B. As a result the subtask i can compute the element c_{ii} of the result matrix C. Further each subtask transmits its column of matrix B to the following subtask in accordance with the ring structure. These actions should be repeated until all the iterations of the parallel algorithm are completed.

2. The second algorithm. The difference of the second algorithm from the first one is that the subtasks contain not columns but rows of matrix B. As a result, data multiplication of each subtask is the multiplication of the row elements of the matrix B by a corresponding row element of the matrix A. Therefore, a row of partial results for matrix C is obtained in each subtask.

In case of this scheme of data decomposition for matrix multiplication, it is necessary to provide sequential obtaining all rows of the matrix B by all in the subtasks, the multiplication of the row elements of the matrix B by a corresponding row element of the matrix A and summation of the new values and the previously computed ones. The ring structure of information dependencies may be also used to provide the necessary sequence of communications of the rows of the matrix B among the subtasks (see Figure 13.3).

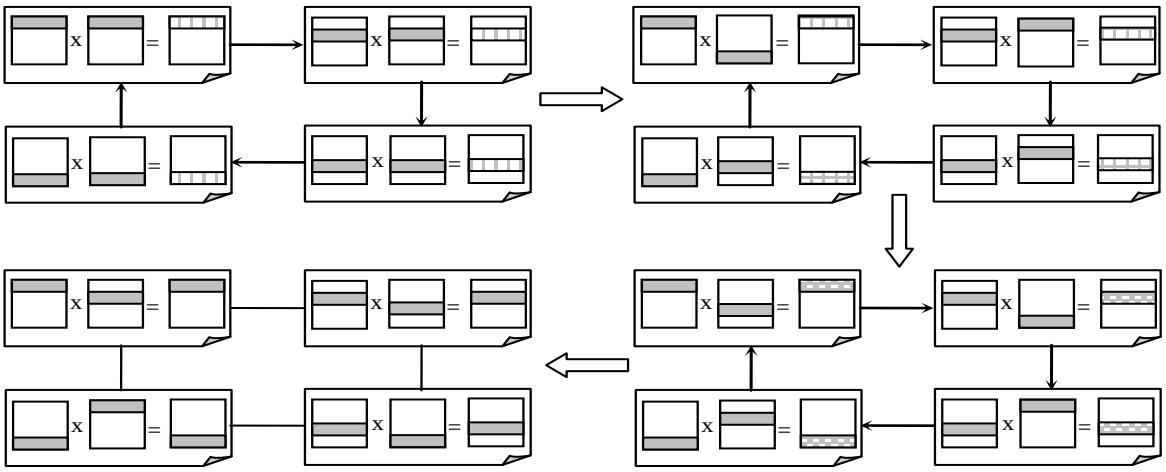

Figure 13.3. General scheme of data communications for the second parallel algorithm of matrix multiplication in case of block-striped decomposition

Figure 13.3 presents the iterations of the matrix multiplication algorithm in the case when matrices have 4 rows and 4 columns (n=4). At the beginning of the computations each subtask i, $0 \le i < n$, holds i-th rows of the matrix A and the matrix B. As a result of multiplication the subtask defines i-th row of the partial results for the matrix C. Then each subtask transmits its row of the matrix B to the following subtask according to the ring structure of information dependencies. The described actions are repeated until all the iterations of the parallel algorithm are completed.

The lecture gives a detailed review of the MPI-based implementation of the methods above.

The results of the computational experiments for first algorithm are shown in Table 13.1. The experiments were performed with the use of 2, 4 and 8 processors.

Table 13.1. 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

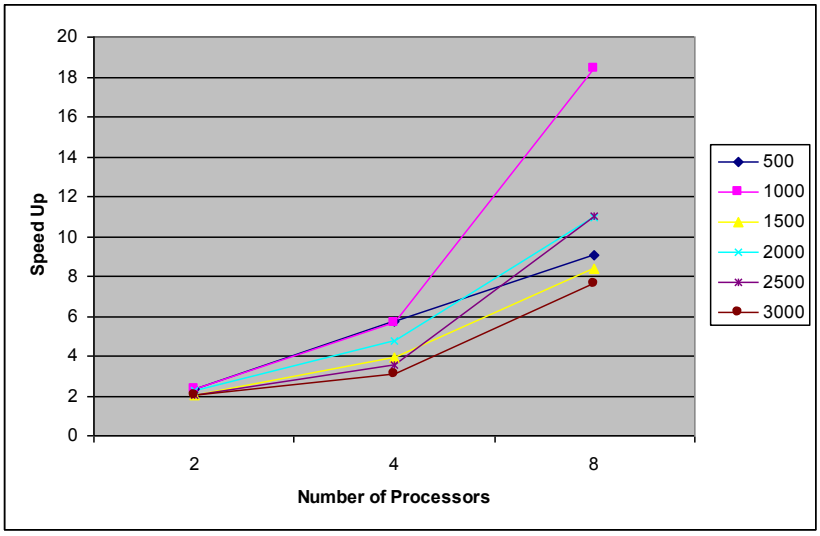

Figure 13.4. Speedup for the first parallel algorithm of matrix multiplication (block-striped matrix decomposition)

Test questions

1. What is the statement of the matrix multiplication problem?
2. Give the examples of the problems, which make use of the matrix multiplication operations.
3. Give the examples of various sequential algorithms of matrix multiplication operations. Is the complexity various in case of different algorithms?
4. What methods of data distribution are used in developing parallel algorithms of matrix multiplication?
5. Describe the general schemes of the parallel algorithms considered in the Section.
6. Analyze and compute the efficiency of the block-striped algorithm for horizontal partitioning of the multiplied matrices.
7. What information communications are carried out for the algorithms in case of the block-striped data decomposition?
8. What information interactions are carried out in case of the checkerboard block matrix multiplication algorithms?
9. What functions of the library MPI appear to be necessary in the software implementation of the algorithms?

Practice

1. Implement two block striped matrix multiplication algorithms. Compare their runtime.

2. Implement the Fox algorithm. Perform computational experiments. Compare experimental results to those of earlier implementations.

References

1. **Dongarra, J.J., Duff, L.S., Sorensen, D.C., Vorst, H.A.V. (1999).** Numerical Linear Algebra for High Performance Computers (Software, Environments, Tools). Soc for Industrial & Applied Math/
2. **Blackford,** L. S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J. J., Hammarling, S., Henry, G., Petitet, A., Stanley, D. Walker, R.C. Whaley, K. (1997). Scalapack Users' Guide (Software, Environments, Tools). Soc for Industrial & Applied Math.
3. **Foster**, I. (1995). Designing and Building Parallel Programs: Concepts and Tools for Software Engineering. Reading, MA: Addison-Wesley.
4. **Andrews**, G. R. (2000). Foundations of Multithreaded, Parallel, and Distributed Programming.. – Reading, MA: Addison-Wesley (русский перевод Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования. – М.: Издательский дом "Вильямс", 2003)
5. **Kahaner, D**., Moler, C., Nash, S. (1988). Numerical Methods and Software. – Prentice Hall (русский перевод Каханер Д., Моулер Л., Нэш С. Численные методы и программное обеспечение. М.: Мир, 2001)
6. **Quinn**, M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.
7. **Wilkinson**, B., Allen, M. (1999). Parallel programming. – Prenrice Hall.

LECTURE 14

Together with representation of matrices as sets of rows and columns, checkerboard matrix representation is also widely used. Lecture 14 is a follow-up of Lecture 13. This lecture describes parallel algorithms based on checkerboard block data decomposition. This is illustrated by the Fox algorithm – a parallel matrix multiplication algorithm.

To develop a parallel matrix multiplication method based on the checkerboard decomposition scheme it should be reminded that in this case the basic subtasks are responsible for computing the separate blocks of the matrix *C*. It is also required that each subtask should hold only one block of the multiplying matrices at each iteration.

To enumerate the subtasks the indices of the blocks C_{ij} contained in the subtasks can be used for enumeration. Thus, the subtask *(i,j)* computes the block C_{ij}. So the set of subtasks forms a square grid, which corresponds to the structure of the checkerboard block decomposition of the matrix C.

The Fox algorithm can be used to perform matrix multiplication computations under these conditions.

In accordance with the Fox algorithm each basic subtasks *(i,j)* holds four matrix blocks:

– Block C_{ij} of matrix *C*, computed by the subtask;

– Block A_{ij} of matrix *A*, placed in the subtask before the beginning of computations;

– Blocks A'_{ij} , B'_{ij} of matrices *A* and *B*, obtained by the subtask in the course of computations.

Parallel algorithm execution includes:

- **The initialization stage**. Each subtask *(i,j)* obtains blocks A_{ij}, B_{ij}. All elements of blocks C_{ij} in all subtasks are set to zero;
- **The computation stage**. At this stage the following operations are carried out at each iteration l, $0 \le l < q$,:
  – For each row i, $0 \le i < q$, the block A_{ij} of subtask *(i,j)* is transmitted to all the subtasks of the same processor grid row; index *j*, which defines the position of the subtask in the row, is computed according to the following expression:

$$j = (i+l) \bmod q,$$

  where *mod* is operation of obtaining the remainder in integer division;:

  – Blocks A'_{ij}, B'_{ij} obtained as a result of subtask communications are multiplied and added to block C_{ij}:

$$C_{ij} = C_{ij} + A'_{ij} \times B'_{ij};$$

  – Blocks B'_{ij} of each subtask *(i,j)* are transmitted to the subtasks, which are upper neighbors in the processor grid columns (the first row blocks are transmitted to the last row of the grid).

To illustrate these rules we show the state of blocks in each subtask in the course of executing iterations of the computation stage (for the grid of 2×2). See Figure 14.1.

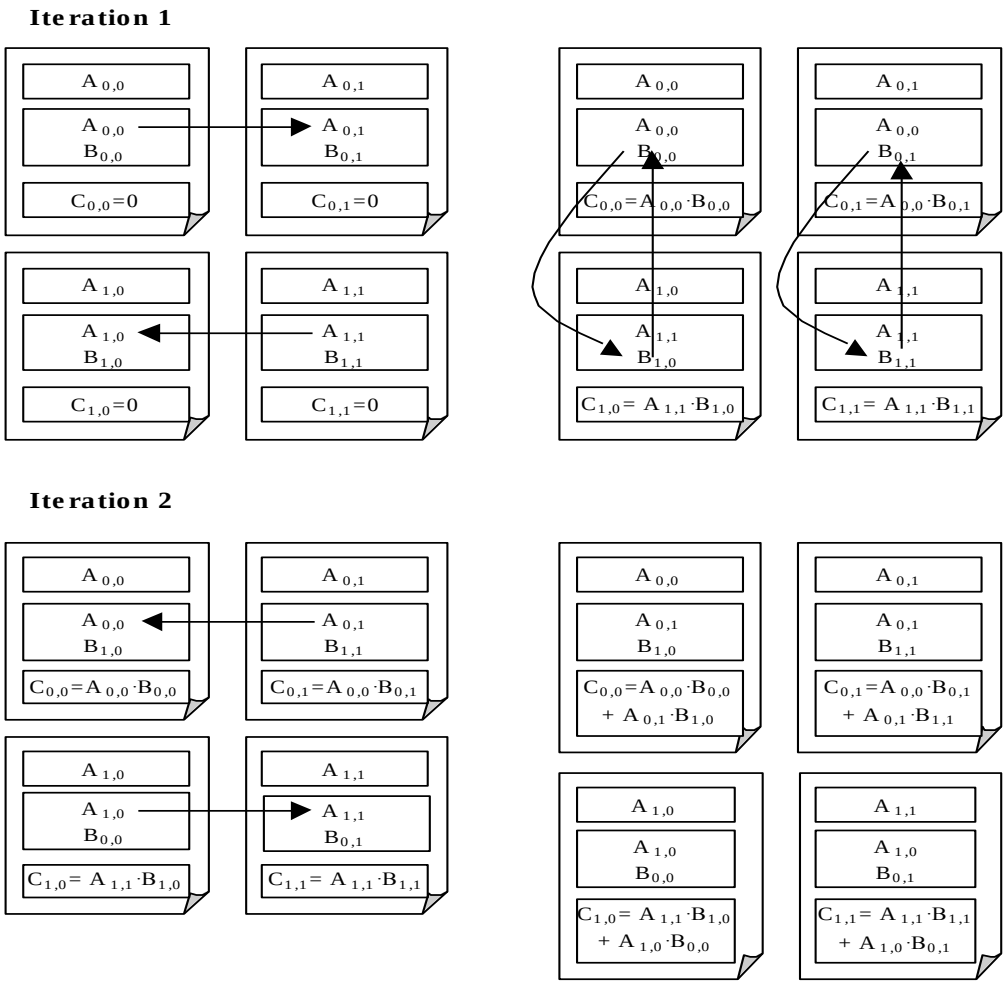

Figure 14.1. Block distribution among subtasks on iterations of the Fox algorithm

The results of the experiments with the use of 4 and 9 processors are given in Table 14.1.

Table 14.1 The Results of the computational experiments for estimating the Fox parallel algorithm efficiency

Matrix Size	Serial Algorithm	Parallel Algorithm			
		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

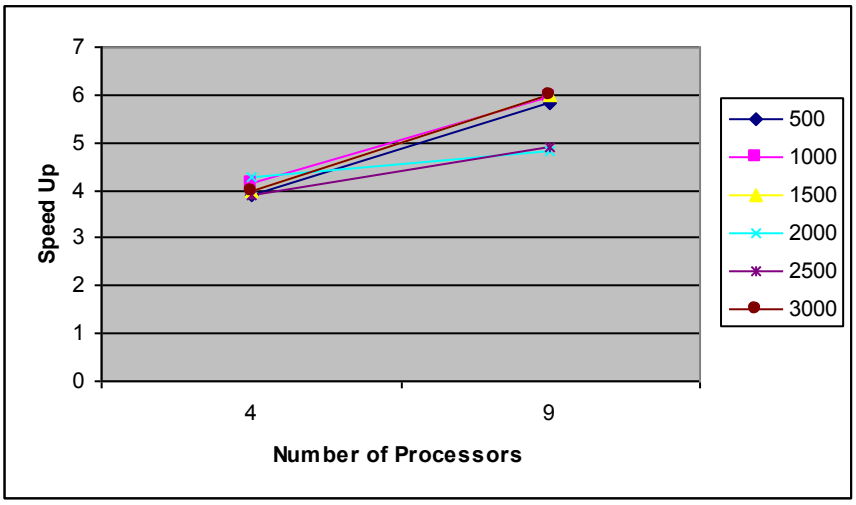

Figure 14.2. Speedup of the Fox Parallel Algorithm with Respect to Number of Processors