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

Lecture 11, 12. Parallel Methods for Matrix-Vector Multiplication for Systems with Distributed Memory.

Nizhni Novgorod

2014

Lectures_11,12_. Parallel Methods for Matrix-Vector Multiplication for Systems with Distributed Memory

This lecture discusses the parallel methods for matrix-vector multiplication. It deals with the issue of sharing the processed matrix by parallel processes. To distribute the load, a library implementing the MPI standard is used.

Let us assume that the matrices, we are considering, are dense, i.e. the number of zero elements in them is insignificant in comparison to the general number of matrix elements.

The result of multiplying the matrix *A* of order $m \times n$ by vector *b*, which consists of *n* elements, is the vector *c* of size *m*, each *i*-th element of which is the result of inner multiplication of *i*-th matrix *A* row (let us denote this row by a_i) by vector *b*:

$$c_i = (a_i, b) = \sum_{j=0}^{n-1} a_{ij} b_j \ , \ 0 \le i \le m-1 . \qquad (11.1)$$

Thus, obtaining the result vector *c* can be provided by the set of the same operations of multiplying the rows of matrix *A* by the vector *b*. Each operation includes multiplying the matrix row elements by the elements of vector *b* (*n* operations) and the following summing the obtained products (*n-l* operations). The total number of necessary scalar operations is the value

$$T_1 = m \cdot (2n - 1) .$$

The sequential algorithm of multiplying matrix by vector may be represented in the following way:

```
// Algorithm 11.1
// Sequential algorithm of multiplying matrix by vector
for (i = 0; i < m; i++){
  c[i] = 0;
  for (j = 0; j < n; j++){
    c[i] += A[i][j]*b[j]
  }
}
```

The first example of a parallel algorithm is the parallel matrix-vector multiplication one based on representation of a matrix as continuous sets of rows (horizontal bands). For such a way of data partitioning one can select the operation of scalar multiplication of one matrix row by a vector.

The general scheme of subtask interaction in the course of computation in shown in Fig. 11.1. The lecture reviews possibilities to implement a MPI-based parallel algorithm.

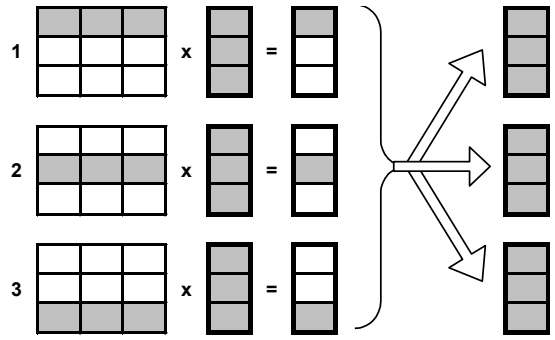

Fig.11.1 Computational scheme for the parallel algorithm of matrix-vector multiplication based on rowwise matrix decomposition

Find below the results of experiments described in the lectures.

See the experimental results in Table 11.1.The experiments were performed with the use of 2, 4 and 8 processors. The algorithm runtime is in seconds.

Table 11.1. Results of computational experiments for the parallel algorithm of matrix multiplication by a vector based on rowwise data decomposition

Matrix size	Sequential algorithm	Parallel algorithm					
		2 processors		4 processors		8 processors	
		Time	Speedup	Time	Speedup	Time	Speedup
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

Test questions

1. What are the main methods of distributing matrix elements among processors?
2. What is the statement of the matrix-vector multiplication problem?
3. What is the computational complexity of the sequential matrix-vector multiplication?
4. Why is it admissible to duplicate the vector-operand to all the processors in developing a parallel algorithm of matrix-vector multiplication?
5. What approaches of the development of parallel algorithms may be suggested?
6. Describe the general schemes of the parallel algorithms discussed in the Section.
7. Evaluate the efficiency characteristics for one of the algorithms discussed in the Section?
8. Which of the algorithms has the best speedup and efficiency?
9. Can the use of the cyclic data distribution scheme influence the execution time of each of the algorithms?
10. What information communications are carried out for the algorithms in case of block-striped data distribution scheme? What is the difference between the data communications in case of rowwise matrix distribution and those required in case of columnwise distribution?

11. What information communications are performed for the checkerboard block matrix-vector multiplication algorithm?
12. What kind of communication network topology is adequate for each algorithm discussed in the Section?
13. What functions of the library MPI appeared to be necessary in the software implementation of the algorithms?

Practice

1. Develop the implementation of the parallel algorithm based on the column wise striped matrix decomposition. Estimate theoretically the algorithm execution time. Carry out the computational experiments.

2. Develop the implementation of the parallel algorithm based on the checkerboard block decomposition. Estimate theoretically the algorithm execution time. Carry out the computational experiments.

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.

LECTURE 12

Lecture 12 is the follow-up of Lecture 11. It describes the second way of data partitioning based on the columnwise matrix decomposition.

In case of columnwise matrix decomposition the operation of multiplying a column of matrix A by one of the vector b elements may be chosen as the basis computational subtask. As a result to perform computations each basic subtask i, $0 \le i < n$, must contain the i-th column of matrix A and the i-th elements b_i and c_i of vectors b and c.

At the starting point of the parallel algorithm of matrix-vector multiplication each basic task i carries out the multiplication of its matrix A column by element b_i. As a result, vector $c'(i)$ (the vector of intermediate results) is obtained in each subtask. The subtasks must further exchange

their intermediate data in order to obtain the elements of the result vector c (element j, $0 \le j < n$, of the partial result $c'(i)$ of the subtask i, $0 \le i < n$, must be sent to the subtask j). This *all-to-all communication* or *total exchange* is the most general communication procedure and may be executed with the help of the function *MPI_Alltoall* of MPI library. After the completion of data communications each basic subtask i, $0 \le i < n$, will contain n partial values $c'_i(j)$, $0 \le j < n$. Element c_i of the result vector c is determined after the addition of the partial values (see Figure 12.1).

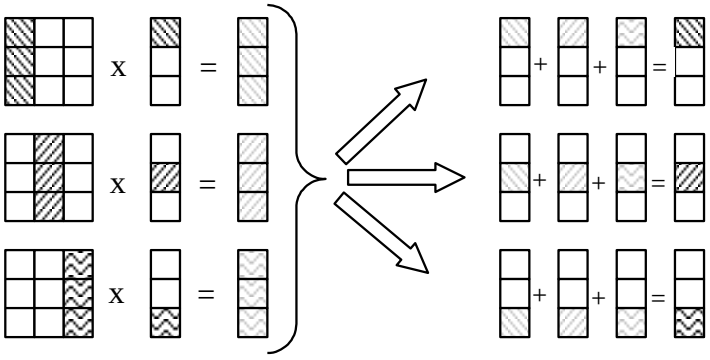

Figure 12.1 Computation scheme for parallel matrix-vector multiplication based on columnwise striped matrix decomposition

The results of the computational experiments are given in Table 12.1.

Table 12.1. The results of the computational experiments for parallel matrix-vector multiplication algorithm based on columnwise matrix decomposition

Matrix Size	Sequential Algorithm	2 processors		4 processors		8 processors	
		Time	Speed up	Time	Speed up	Time	Speed up
1000	0,0041	0,0022	1,8352	0,0132	0,3100	0,0008	4,9409
2000	0,016	0,0085	1,8799	0,0046	3,4246	0,0029	5,4682
3000	0,031	0,019	1,6315	0,0095	3,2413	0,0055	5,5456
4000	0,062	0,0331	1,8679	0,0168	3,6714	0,0090	6,8599
5000	0,11	0,0518	2,1228	0,0265	4,1361	0,0136	8,0580

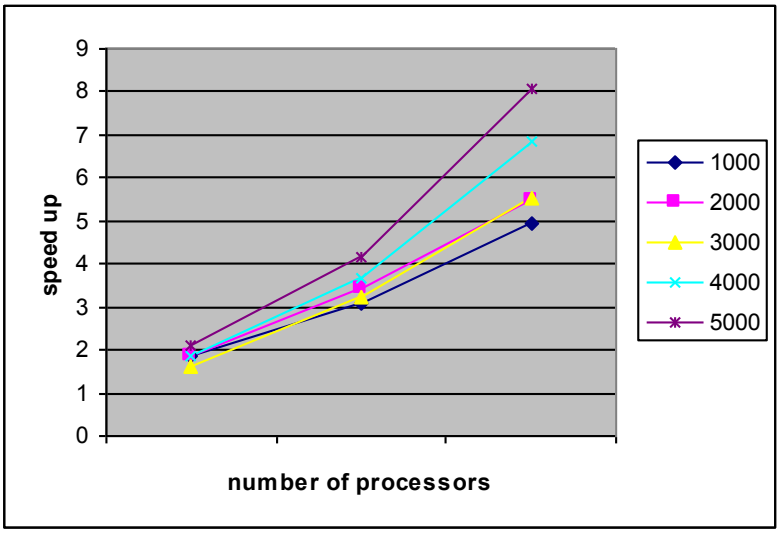

Figure 12.1. Speedup for parallel matrix-vector multiplication (columnwise block-striped matrix decomposition)