|  |  |
| --- | --- |
|  |  |

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”

**DIRECT METHODS   
FOR SOLVING SYSTEM OF LINEAR EQUATIONS**

*Lecture 1. Introduction. Implementation of the standard matrix algorithms  
in shared memory systems.*

Nizhni Novgorod

2014

# Objectives

An objective of the lecture is investigation of the standard matrix algorithms (by example of matrix-vector and matrix-matrix multiplication) and also study of the approaches to their efficient parallel implementation in shared memory systems.

# Abstract

The basic matrix algorithms are considered (matrix-vector and matrix-matrix multiplication). The matrices are supposed to be dense. In order to organize parallel calculation the different matrix decomposition strategies are examined: rowwise, columnwise, blockwise. The estimates for algorithms complexity are provided. The most important elements are an estimate for the cache-memory impact and constructing the algorithms that use cache memory efficiently.

# guidelines

The matrix computations are a base for a lot of scientific and engineering calculations. Being computationally consuming, the matrix computations represent a classical field of parallel computations application.

A lot of methods for matrix computations are characterized with repetition of the same computational actions for the different matrix elements. This indicates the presence of *data parallelism* when performing matrix calculations, and, as a result, in most cases, matrix operations parallelizing reduces to distributing the processed matrices among threads. Choosing the way of matrix distribution, one defines the specific method of parallel calculations; existence of the various schemes for data distribution generates a range of *parallel algorithms of matrix computations*. The most common and widely used methods for matrix distribution are decomposing the data into *stripes* (vertical or horizontal) or into rectangular fragments (*blocks*).

At the beginning, the matrix-vector multiplication is examined. A definitive description for the algorithm, the estimate of its complexity and also its sequential implementation are given. The most appropriate matrix distribution for the parallel implementation of matrix-vector multiplication is block-striped horizontal decomposition. In this case a certain subset of matrix rows is assigned to each thread. For this algorithm sequential and parallel implementations are stated and also the algorithms comparison is made in order to determine parallel version speedup. The results of the experiments confirm the adequacy of constructed estimates for the algorithm complexity.

After that, matrix-vector multiplication is considered. At first, a definitive description for the algorithm is stated as well, the complexity estimate and sequential implementation of the algorithm are given. Then a basic parallel algorithm for the matrices multiplication (without any distribution) is examined. Its performance is estimated taking into account cache-memory use. The results of the experiments are stated. Insufficient efficiency of the cache use in this algorithm is demonstrated.

One way to solve the problem of efficient cache-memory use is to decompose the matrices involved in the algorithm into blocks. A description for block algorithm for matrices decomposition and the estimates for its performance subject to cache-memory use are given.

The results of the experiments demonstrating the adequacy of constructed estimates are shown. However, it is necessary to specify a block size when dividing the matrices so that cache-memory is used most efficiently. The concrete recommendations on how to make such choice are given (for both sequential and parallel algorithms). For the case of block algorithm for matrices multiplication with the optimal block size, the results of computing experiments are also demonstrated. These results indicate a considerable speedup in comparison with both sequential and parallel algorithms.

# RECOMMENDATIONS for students

Matrix multiplication problems are often used as a demonstrative example in parallel programming. Therefore, they are widely used in the literature. The papers [1], [2] may be recommended as an additional educational material. The wide discussion of the parallel realization of matrix computations is given in [3].

# References

1. Kumar V., Grama A., Gupta A., Karypis G. Introduction to Parallel Computing. Addison Wesley, 2003.
2. M. Quinn. Parallel programming in C with MPI and OpenMP. McGraw-Hill, 2004.
3. Dongarra J.J., Duff I.S., Sorensen D.C., van der Vorst H.A. Numerical Linear Algebra for High-Performance Computers. SIAM, 1999.

# EXERCISES

1. Implement the parallel matrix-vector multiplication algorithm based on block matrix decomposition. Derive theoretical estimates of the time consumed by the algorithm taking into account the parameters of the used computing system. Carry out the computing experiments. Compare the results of real experiments with the theoretical estimates derived before.
2. Design the implementation of block algorithms for matrices multiplication that might be executed for rectangular grids of the threads of general form.

# TEST QUESTIONS

1. What is the complexity order for matrix-vector multiplication?
   1. O(n)
   2. +O(n2)
   3. O(n3)
2. What is the complexity order for matrix-matrix multiplication?
   1. O(n)
   2. O(n2)
   3. +O(n3)
3. For the efficient implementation of parallel matrix-vector multiplication algorithm based on block matrix decomposition, it is necessary that the block sizes would:
   1. (+) be approximately equal to the size of computing core cache,
   2. be smaller in height than in width,
   3. fit in width to the cache-line
4. For the efficient implementation of parallel matrix-vector multiplication algorithm based on block-striped horizontal matrix decomposition, it is necessary that the stripe height would:
   1. be as greater as possible,
   2. (+)no matter in the case of great data volume,
   3. be as greater as possible in case of small matrices and be as smaller as possible in case of great ones
5. When performing the parallel algorithm based on block-striped horizontal matrix decomposition, the collection of resulting vector data is made using:
   1. an operation of data transmission from one core to another,
   2. (+) shared memory used by all threads,
   3. an operation of data reduction on zero core
6. What schemes of data distribution might be used when developing parallel algorithms of matrices multiplication?
   1. (+) block-striped data decomposition,
   2. (+) block data decomposition,
   3. data are duplicated among processors
7. What scheme of data distribution is the most efficient when developing parallel algorithms of matrices multiplication?
   1. block-striped data decomposition,
   2. (+) block data decomposition,
   3. data are duplicated among processors
8. For the efficient implementation of parallel matrix-matrix multiplication algorithm it is necessary to use the algorithm of:
   1. (+) block data decomposition with the small blocks sizes,
   2. block data decomposition with the large blocks sizes,
   3. rowwise data decomposition
9. For the efficient implementation of parallel matrix-matrix multiplication algorithm using block data decomposition it is necessary that:
   1. a number of blocks would equal a number of processor cores,
   2. block size would be three times greater than cache volume,
   3. (+) a volume of data in block would be equal to cache volume
10. Efficient implementation of block algorithm for multiplication of matrices with small block sizes is ensured by:
    1. the absence of cache misses,
    2. (+) a small number of cache misses,
    3. a great number of cache misses