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

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**

*Practice 2. Sparse matrix-vector multiplication. Loop parallelization   
with OpenMP, TBB, ArBB, Cilk Plus.*

Nizhni Novgorod

2014

# Objectives

The objective of this work is to investigate some parallel programming technologies in shared memory systems by example of sparse matrix-vector multiplication problem.

# Abstract

Threads creation could be considered as a basic way to achieve parallelism in applications developed for shared memory systems. Multithreading provides core-level parallelism. Threads can solve both one-type problems for the different input data sets and different-type problems. Therefore, the problems associated with load balancing among threads, threads interaction organization, threads synchronization, etc., arise. Using high-level interfaces and libraries for working with threads partly automates the solving of such questions, but does not relieve the developer of responsibility for the results correctness. At present moment, multithreading model is supported, in particular, by such technologies as OpenMP, Intel® Threading Building Blocks (Intel® TBB), Intel® Cilk Plus and Intel® Array Building Blocks (Intel® ArBB).

At given work the base opportunities of the technologies mentioned above will be studied. Parallelization of loops with a known number of repetitions will be mainly taken into consideration since this problem is the most widely spread on practice. Investigation of more complex situations is a question of other laboratory works.

# GUIDELINES

In the introduction the notion of a sparse matrix is considered, this notion is used in the different sources and discussed in the fields where the sparse algebra problems arise. Then, the problem of multiplication of a sparse matrix *A (n*×*n)* by a dense vector *b* with elements of **double** type is stated.

The Compressed Row Storage format (CRS) for sparse matrices and its modifications are considered. The CRS format is used when implementing the algorithms.

Then the question of choosing the test problems so that they provide the result reproducibility and representativeness is discussed. A generator for a class of the matrices containing at the most *K* randomly placed nonzero elements in every row is used in the work.

The function **mkl\_dcsrgemv()**of the MKL library is interpreted as a model version for matrix-vector multiplication. The description of the sequential version ant its comparison with the model one is stated. Some lag from the MKL version is demonstrated.

After that, the parallel version for the program using the OpenMP technology is developed. It is simply obtained with the help of a slight modification of the sequential version. Let us organize the rowwise load distribution among the threads that corresponds the outer loop of the sequential implementation. Actually, the loop with a known number of repetitions is to be parallelized. The directive #pragmaompparallelfor is used for this. The option num\_threads is additionally specified in order to further control the number of created threads and conduct a scalability analysis. The experiments demonstrate good scalability of the parallel program.

Next, the TBB-version of the parallel program is designed. An easy code modification is not enough when developing the TBB-version as opposed to the OpenMP-version. The function tbb::parallel\_for is used for parallelizing the loops with a known number of repetitions. This function takes an iteration space and instance of a function class as input parameters. The standard one-dimensional iteration space realized in the TBB library will be used as the iteration space. Let us develop the function class. This class contains the following fields: the CRS-format matrix, vector by which the matrix is multiplied and resulting vector. The constructor and the method operator() must be implemented in the class. The last one takes the iteration space as an argument. In whole, the execution times for the TBB- and OpenMP-versions are close if the number of the threads is the same. Moreover, the same effect is observed – the one-thread version is almost 2 times slower that the sequential one.

The next parallel implementation will be developed using the CilkPlus technology. It is enough to apply the Intel compiler and specify the necessary project settings. The syntax construct cilk\_for is used in CilkPlus for parallelizing the loops with a known number of repetitions. In essence, the parallel version is obtained from the sequential one substituting for for cilk\_for. The experiments results demonstrate good scalability of the program.

The last parallel implementation will be designed using the Intel Array Building Blocks (ArBB) library. Let us begin with developing the algorithm since the ArBB parallelism model makes a start from the data, but not from the physical system cores. In essence, it is necessary to understand how to move from the scalar operations to the vector ones. As for the dense matrix-vector multiplication, it is actually a set of scalar matrix row-vector multiplications. Scalar multiplication stands for componentwise vectors multiplication and further addition. Thus, two vector operations are obviously distinguished – the componentwise vectors multiplication and addition of the vector components. The multiplication of the sparse matrix is almost indistinguishable from the case of the dense one. The only difference is that the scalar multiplication is calculated not for the whole matrix row but only for the vector containing the nonzero elements of the row and the corresponding elements of the specified vector. On the other hand, since the CRS format is used for the sparse matrix, the collection of the nonzero elements is represented by a vector. Let us converse the dense vector by that the matrix is multiplied so that the components corresponding to the numbers of the columns containing the nonzero elements are duplicated in it. Then the matrix-vector multiplication operation can be considered as the scalar multiplication of two vectors with further addition of the vector parts that correspond to the rows of the matrix. It is proposed to use this idea when developing the parallel implementation using the ArBB library.

The experiments results demonstrate that the ArBB-version execution time considerably exceeds the execution time of the other versions. This observation can be explained, first of all, by the fact that the formation of the agreement and the compilation of the ArBB-function is performed when the ArBB-function is called using the operation call() for the first time during the execution. As a consequence, the overhead costs on performing this procedure arise. The execution time is considerably reduced when calling the ArBB-function repeatedly.

# RECOMMENDATIONS for students

Notwithstanding the long-standing edition, the book [1] has not lost its relevance today. The main sparse matrix storage formats allowing the efficient implementation of the standard matrix algorithms are considered in the book. Particularly, the sparse matrix-vector multiplication problem is discussed in detail.

The technological basis of the laboratory work is described in [2-13]. Most of the materials are freely available on the Internet.

# References

1. Sergio Pissanetzky. Sparse matrix technology. Academic Press, 1984.
2. Intel® Threading Building Blocks. Reference Manual. Version 1.6. Intel® Corporation, 2007.
3. Intel® Threading Building Blocks. Tutorial. Version 1.6. Intel® Corporation, 2007.
4. Official Website OpenMP [www.openmp.org].
5. TBB library page on the Intel Corporation Website [http://software.intel.com/en-us/articles/intel-tbb/].
6. CilkPlus interface page on the Intel Corporation Website [http://software.intel.com/en-us/articles/intel-cilk-plus/].
7. Intel® CilkPlus guide [http://software.intel.com/sites/products/evaluation-guides/docs/cilk-plus-evaluation-guide.pdf].
8. Intel® CilkPlus syntax constructs specification [http://software.intel.com/sites/products/cilk-plus/cilk\_plus\_language\_specification.pdf].
9. Intel® CilkPlus documentation [http://software.intel.com/sites/products/documentation/studio/composer/en-us/2011/compiler\_c/index.htm#cref\_cls/common/cilk\_for.htm].
10. ArBB library page on the Intel® Corporation Website [http://software.intel.com/en-us/articles/intel-array-building-blocks/].
11. Brief description of the Intel® Array Building Blocks library [http://drdobbs.com/go-parallel/article/showArticle.jhtml?articleID=227300084].
12. Article Array Building Blocks: A Flexible Parallel Programming Model for Multicore and Many-Core Architectures [http://software.intel.com/sites/products/documentation/arbb/arbb\_manual\_win].
13. MKL library page on the Intel® Corporation Website [http://software.intel.com/en-us/articles/intel-mkl/].

# EXERCISES

1. Develop parallel implementation of the dense matrices multiplication algorithm using the technologies OpenMP, Cilk and TBB.
2. Design two implementations for the scalar multiplication of the dense vectors using the functions call and map of the ArBB library.
3. Develop the dense matrices multiplication algorithm using the ArBB library. Remark: consider the matrices multiplication problem as the repeated calculation of the scalar products of the first matrix rows and the second matrix columns.
4. Estimate the number of cache misses when increasing the number of the threads in the OpenMP-implementation in order to find out the reasons for the nonlinear scalability. Use the instrument Intel Parallel Studio XE.
5. Explain the reason for the absence of the linear speedup of the parallel implementations developed with OpenMP, Cilk and TBB.
6. Design a generator for the matrices of any other structure (e.g., with the increasing number of the nonzero elements) and carry out the experiments with all developed implementations.

# TEST QUESTIONS

1. In what positions of the array **Values** the elements of the **i**-th row of the matrix are stored if the matrix is stored in the CRS format?
   1. From **Columns[i]** to **Columns[i + 1] - 1**
   2. + From **RowIndeх [i]** to **RowIndeх[i + 1] - 1**
   3. From **RowIndeх [i]** to **RowIndeх[i + 1]**
2. What construct of the OpenMP technology is used for parallelizing loops?
   1. +#pragma omp parallel for
   2. #pragma omp parallel
   3. #pragma omp for
3. What parameters are needed for the OpenMP directive **for** in order to distribute the loop iterations among the threads into the blocks of size **chunk** before the start of the loop?
   1. **schedule(chunk, static)**
   2. **schedule(dynamic, chunk)**
   3. + **schedule(static, chunk)**
4. What is passed as an argument to the method **operator()** of the function class in TBB?
   1. nothing
   2. + a loop iteration space
   3. a number of the loop iteration
5. What is the method initialize(…) of the tbb::task\_scheduler\_init class responsible for?
   1. It allows to set the minimum grainsize when partitioning the iteration space during the loop parallelization.
   2. + It activates an object of the tbb::task\_scheduler\_init class, also its optional parameter allows to set the threads number.
   3. It allows to set the threads number.
6. If the deactivate method terminate() is not called explicitly for an object of the tbb::task\_scheduler\_init class, then
   1. +the object will be deactivated at the moment of calling of a destructor of the corresponding object.
   2. the object will not be deactivated.
   3. the object will be deactivated at the moment when the parallel section of the program completes.
7. How can one set the number of workers in a parallel program using the capabilities of Intel® CilkPlus?
   1. With the help of the function \_\_cilkrts\_set\_param(…) within the program.
   2. With the help of the environment variable CILK\_NWORKERS.
   3. With the help of the corresponding option in the Microsoft Visual Studio development environment.
   4. +All of the listed methods are possible.
8. Is it possible to use the Intel® CilkPlus extension of the С/С++ languages in the absence of the Intel compiler?
   1. +Yes.
   2. No.
9. What type of parallelism is provided by the Intel® Array Building Blocks library?
   1. +Core-level parallelism and data-level parallelism.
   2. Data-level parallelism.
   3. Task-level parallelism.
10. What functionality is provided by the function reshape\_nested\_offsets(…)?
    1. The function provides repacking of the elements of an irregular container.
    2. + The function provides partitioning a regular container into the vectors of different length.
    3. The function provides partitioning a regular container into the vectors of the same length.