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 10. Parallel Programming with MPI.

Collective Data Transmission Operations.

Nizhni Novgorod

2014

Lecture_10_. Parallel Programming with MPI. Collective Data Transmission Operations

This lecture is dedicated to MPI-based parallel programming methods for distributed memory systems. While Lecture 3 described the minimum required set of functions for MPI-based software development, this one describes a number of collective operations ensuring a more efficient data exchange. It also gives a program example demonstrating a way to handle collective operations.

The functions *MPI_Send* and *MPI_Recv*, discussed in lecture 9, provide the possibility of executing data passing operations between two parallel program processes. To execute collective communication operations characterized by participating of all the processes in a communicator, MPI provides a special set of functions. This subsection discussed three functions, which are widely used even in the development of comparatively simple parallel programs. Collective operations are discussed in this lecture.

To demonstrate the example of MPI function applications we will use the problem of summing up vector *x* elements

$$S = \sum_{i=1}^{n} x_i .$$

The development of parallel algorithm for solving this problem is not complicated. It is necessary to divide the data into equal blocks, to transmit these blocks to the processes, to carry out the summation of the obtained data in the processes, to collect the values of the computed partial sums on one of the processes and to add the values of partial sums to obtain the total result of the problem. In further development of the demonstrational programs this algorithm will be simplified. All the vector being summed and, not only separate blocks of the vector, will be transmitted to the program processes. Then, the mentioned training task will demonstrate a number of collective operations described as part of this lecture.

In the first part of the lecture is devoted to *collective data transmission operations*. The sequence of the description corresponds to the order, in which the communication operations are discussed in lecture 9. The conclusion given in the subsection says that MPI provides maintaining practically all the basic information exchanges among the processes.

In the second part of the lecture describes the possibilities of MPI in connection with the use of *virtual topologies*. The following MPI supported topologies are described in the subsection: the *rectangular grid* of arbitrary dimension (*Cartesian topology*) and the *graph* topology of any required type.

The final part of this lecture explains the use of collective operations for solving the problem of π numerical computation.

There are a number of sources, which provide information about MPI. First of all, this is the internet resource, which describes the standard MPI: http://www.mpiforum.org. One of the most widely used MPI realizations, the library MPICH, is presented on www.mpich.org.

The following works may be recommended: Group, et al. (1994), Pacheco (1996), Snir, et al. (1996), Group, et al. (1999a). The description of the standard MPI-2 may be found in Group, et al. (1999b).

We may also recommend the work by Quinn (2003), which described a number of typical problems of parallel programming for the purpose of studying MPI. These are the problems of matrix computations, sorting, graph processing etc.

Test questions

1. What is the difference between point-to-point and collective data transmission operations?
2. Which MPI function provides transmitting data from a process to all the processes?
3. What is the data reduction operation?
4. In what cases should we apply barrier synchronization?
5. What is a deadlock? In what cases does the function of the simultaneous transmission/reception guarantee the absence of deadlock situations?
6. What collective data transmission operations are supported in MPI?
7. What is a communicator in MPI?
8. What can new communicators be created for?
9. What is a virtual topology in MPI?
10. What types of virtual topologies are supported in MPI?
11. What may virtual topologies appear to be useful for?

Practice

1. Develop a program for finding the minimum (maximum) vector element value using collective operations.

2. Develop a program for finding the scalar product of two vectors using collective operations.

3. Develop a program for definite integral computation using the method of rectangles.

$$y = \int_a^b f(x)\,dx \approx h \sum_{i=0}^{N-1} f_i, f_i = f(x_i), x_i = i\,h, h = \frac{b-a}{N}$$

References

1. Pacheco, P. (1996). Parallel Programming with MPI. - Morgan Kaufmann.
2. Gropp, W., Lusk, E., Skjellum, A. (1999a). Using MPI - 2nd Edition: Portable Parallel Programming with the Message Passing Interface (Scientific and Engineering Computation). - MIT Press.
3. Gropp, W., Lusk, E., Thakur, R. (1999b). Using MPI-2: Advanced Features of the Message Passing Interface (Scientific and Engineering Computation). - MIT Press.
4. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J. (1996). MPI: The Complete Reference. - MIT Press, Boston, 1996.