Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (ITMO University)

Report

**about laboratory works**

**Assignment 6.**

**Assignment 7.**

**Assignment 8.**

**Student** Pogrebnoy D.A. j4132c

Saint-Petersburg, 2021

# Assignment 6.

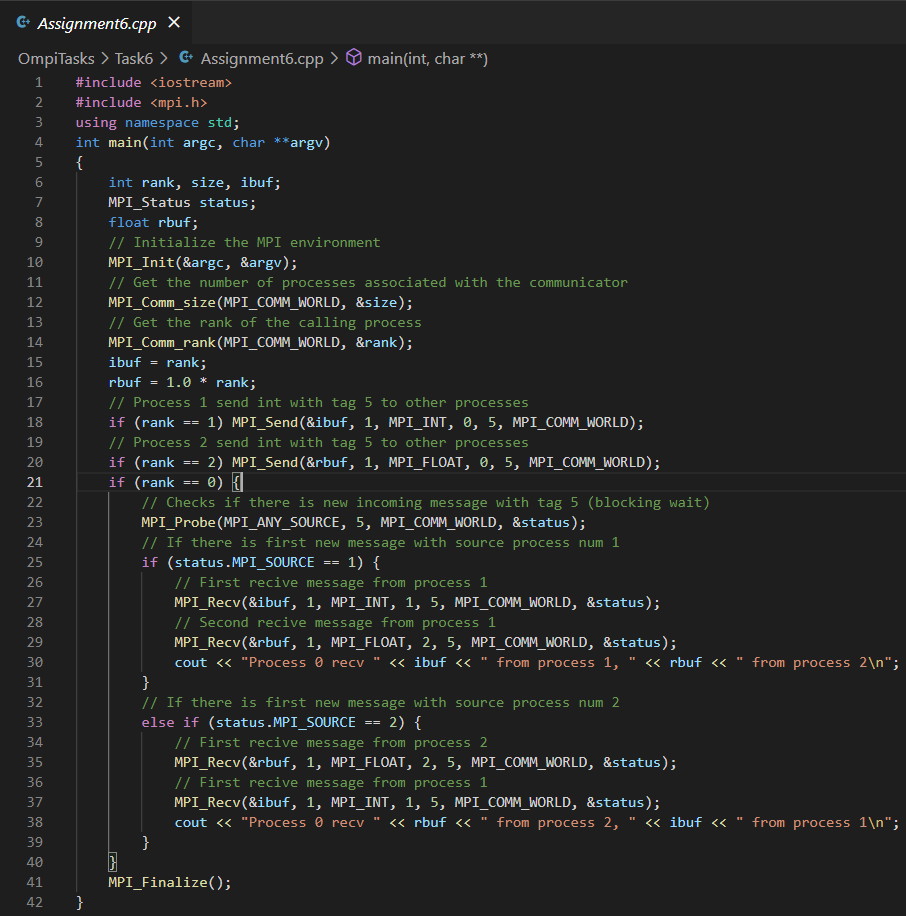
**Task**

Compile the example Assignment6.c in detail, run it and explain it. Transform the program using the MPI\_TAG field of the status structure in the condition.

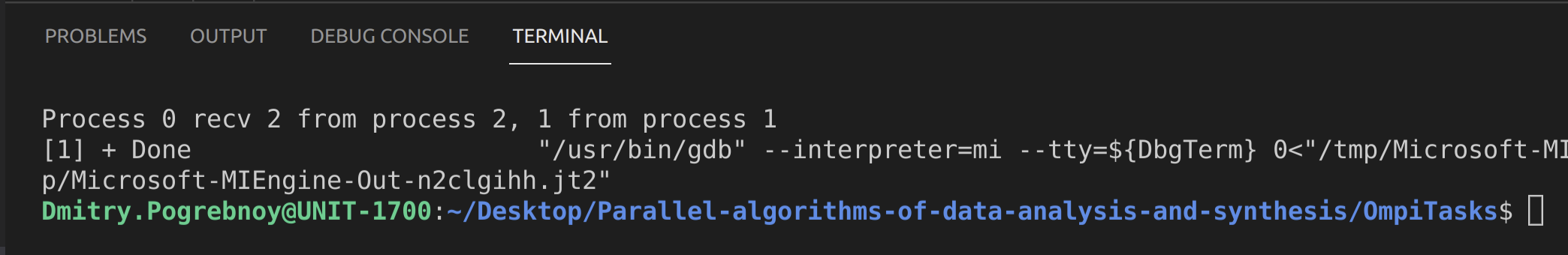
**Implementation**

Source code and data gathered are available on <https://github.com/DmitryPogrebnoy/Parallel-algorithms-of-data-analysis-and-synthesis/tree/master/OmpiTasks/Task6>

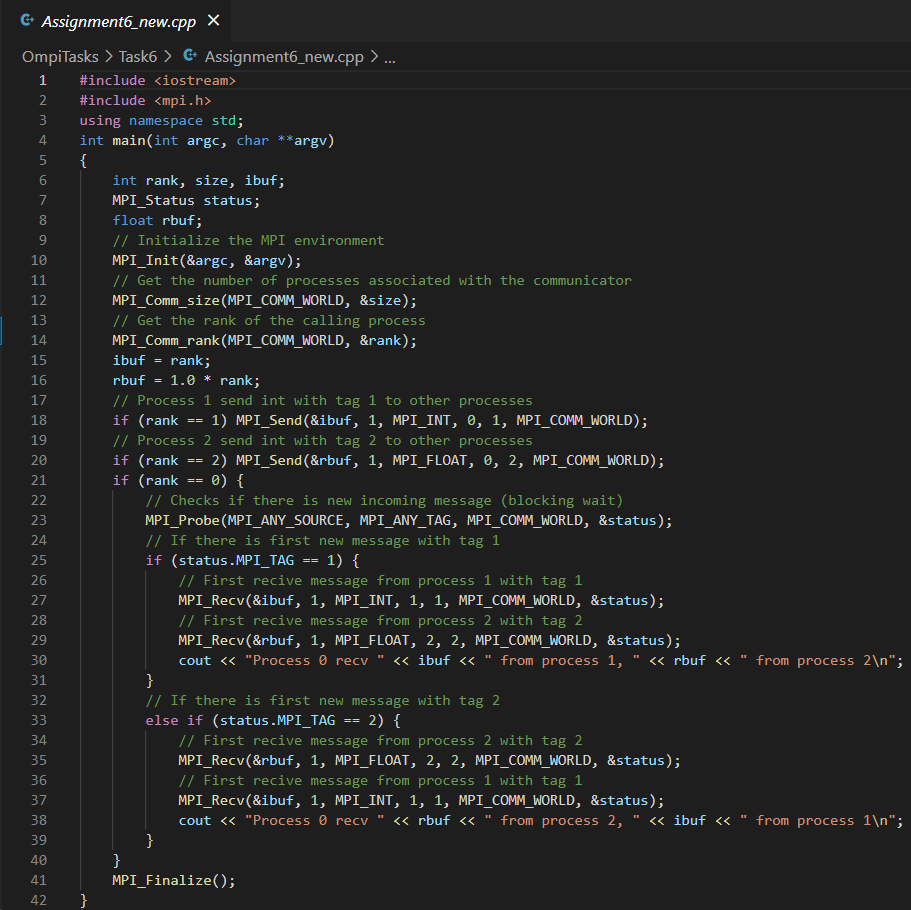
The description of the code is described in the comments.



Output example:



And rewritten version with tags looks like this:



# Assigments 7.

**Task**

Write an MPI program that implements the dot product of two vectors distributed between processes. Two vectors with a size of at least 1,000,000 elements are initialized at process zero and filled with “1”, then they are sent in equal parts to all processes. Parts of vectors are scalar multiplied on each process, the result is sent to the root process and summed up. The total is displayed.

**Implementation**

Source code and data gathered are available on <https://github.com/DmitryPogrebnoy/Parallel-algorithms-of-data-analysis-and-synthesis/blob/master/OmpiTasks/Task7/Task7.cpp>

The description of the code is described in the comments. In the main process, we initialize the value of the number of array elements and the arrays themselves. Next, we send the number of elements in arrays to all threads. After that, we send a piece of two source arrays to each thread to perform the scalar product of these pieces. After that, the values of each pieces are summed up in the main process and the answer is output. It is important that the length of arrays is entirely divided by the number of processes.

Output example:



# Assignment 8.

**Task**

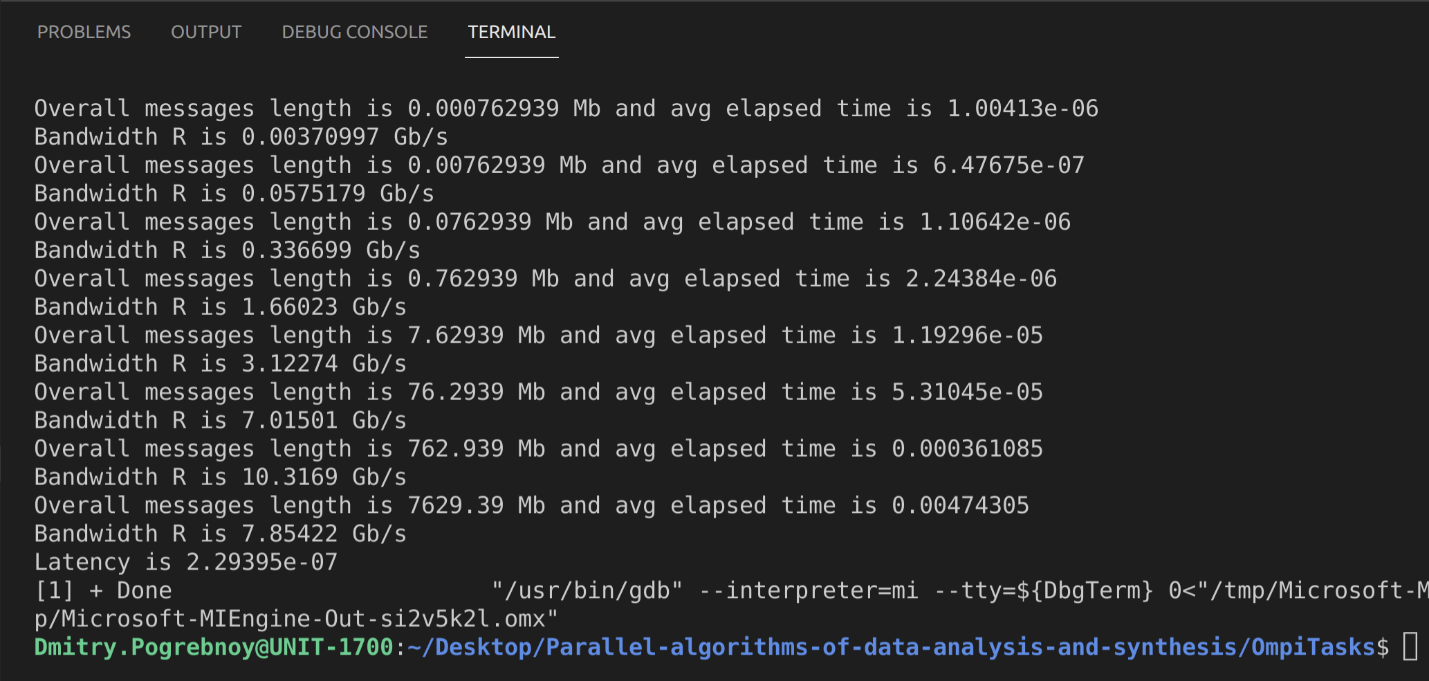
Write an MPI program in which two processes exchange messages, measure the time per exchange iteration, and determine the dependence of the exchange time on the message length. Determine the latency and maximum achievable bandwidth of the communication network. Print the message length in bytes and the throughput in MB/s to the console. Change the length of the message in a loop starting from 1 element and increase to 1,000,000 elements, increasing by 10 times at each iteration.

**Implementation**

Source code and data gathered are available on [https://github.com/DmitryPogrebnoy/Parallel-algorithms-of-data-analysis-and-synthesis/blob/master/OmpiTasks/Task8/Task8.cpp](https://github.com/DmitryPogrebnoy/Parallel-algorithms-of-data-analysis-and-synthesis/blob/master/OmpiTasks/Task8/Task8.cpp%20) .

The description of the code is described in the comments. The program uses two processes. In the main process, we initialize an array with the required length and forward this array 100 times to another process and back, while measuring the time. After that, we measure the delay in sending the message.

Output example:



So maximum achievable bandwidth is ~10 GB/s.