FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION

ITMO UNIVERSITY

Report

MPI. Assignments 9, 10, 11

Parallel algorithms for the analysis and synthesis of data

Performed by
Aleksandr Shirokov
J4133c
Accepted by
Petr Andriushchenko
Deadline: 21.12.21

St. Petersburg
2021

Contents

1 Assignments

1.1 Assignment 9. MPI_Reduce.

1.1.1 Formulation of the problem

1.1.2 Example of launch parameters and output. Detailed description of solution

Code for **assignment 9** is here.
Compilation example: mpic++ -o ./cpf/8.o Assignment8.c
Launch example: mpirun –oversubscribe -np 2 ./cpf/8.o
Let's move to the the code and explain how it works.

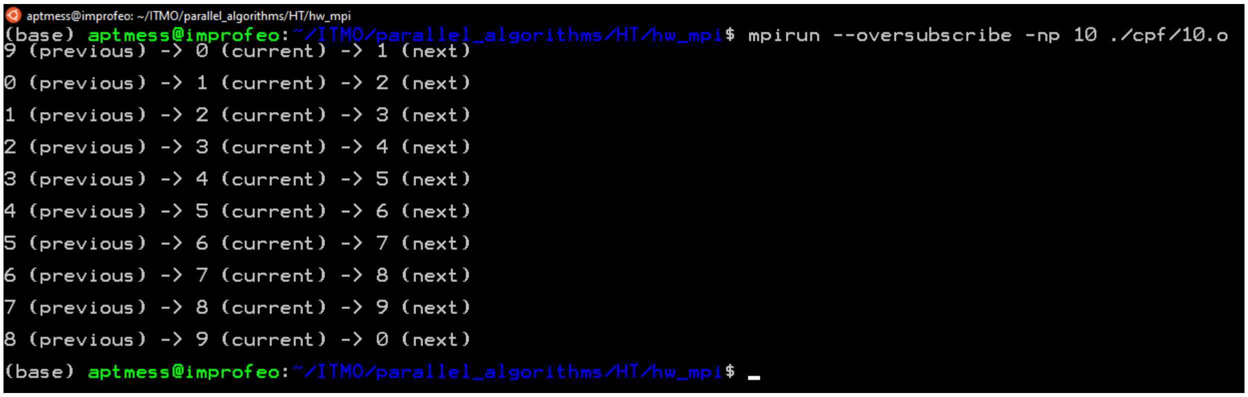
1.2 Assignment 10. MPI. Sending and receiving messages without blocking. Ring exchange using non-blocking operations.

1.2.1 Formulation of the problem

Complete the program Assignment10.c. Compile and run it.
Study the code carefully and explain how it works.

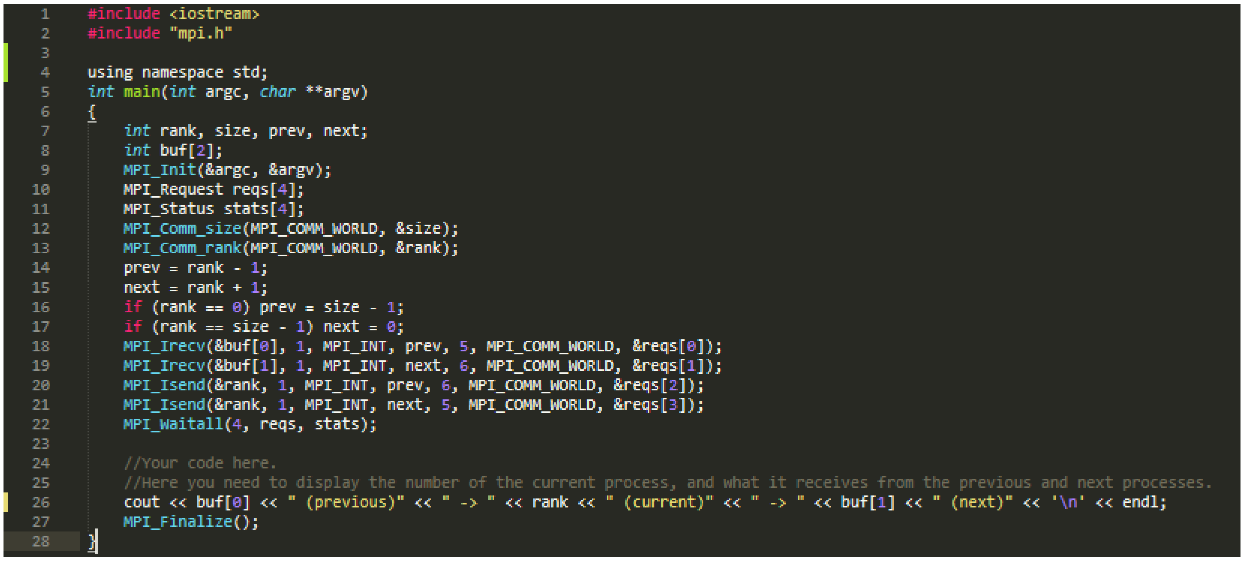
1.2.2 Example of launch parameters and output. Detailed description of solution

Code for **assignment 10** is here.
Compilation example: mpic++ -o ./cpf/10.o Assignment10.c
Launch example: mpirun –oversubscribe -np 10 ./cpf/10.o



```
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ mpirun --oversubscribe -np 10 ./cpf/10.o
9 (previous) -> 0 (current) -> 1 (next)

0 (previous) -> 1 (current) -> 2 (next)

1 (previous) -> 2 (current) -> 3 (next)

2 (previous) -> 3 (current) -> 4 (next)

3 (previous) -> 4 (current) -> 5 (next)

4 (previous) -> 5 (current) -> 6 (next)

5 (previous) -> 6 (current) -> 7 (next)

6 (previous) -> 7 (current) -> 8 (next)

7 (previous) -> 8 (current) -> 9 (next)

8 (previous) -> 9 (current) -> 0 (next)

(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ _
```

Let's move to the the code and explain how it works.



```cpp
#include <iostream>
#include "mpi.h"

using namespace std;
int main(int argc, char **argv)
{
    int rank, size, prev, next;
    int buf[2];
    MPI_Init(&argc, &argv);
    MPI_Request reqs[4];
    MPI_Status stats[4];
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    prev = rank - 1;
    next = rank + 1;
    if (rank == 0) prev = size - 1;
    if (rank == size - 1) next = 0;
    MPI_Irecv(&buf[0], 1, MPI_INT, prev, 5, MPI_COMM_WORLD, &reqs[0]);
    MPI_Irecv(&buf[1], 1, MPI_INT, next, 6, MPI_COMM_WORLD, &reqs[1]);
    MPI_Isend(&rank, 1, MPI_INT, prev, 6, MPI_COMM_WORLD, &reqs[2]);
    MPI_Isend(&rank, 1, MPI_INT, next, 5, MPI_COMM_WORLD, &reqs[3]);
    MPI_Waitall(4, reqs, stats);

    //Your code here.
    //Here you need to display the number of the current process, and what it receives from the previous and next processes.
    cout << buf[0] << " (previous)" << " -> " << rank << " (current)" << " -> " << buf[1] << " (next)" << '\n' << endl;
    MPI_Finalize();
}
```

Assignment 10

The overall goal of the program is that all processes exchange messages with their nearest neighbors (on the left - previous, on the right - next) in accordance with the topology of the ring.

Witg MPI_Waitall the execution of the process is blocked until all exchange operations on the specified reqs identifiers (lines 18-21) are completed and if the error exists in this operations, then the error field in the stats array elements will be set to the appropriate value. In lines $18-21$ there are operations MPI_Irecv and MPI_Isend which are equal to previous functions MPI_Recv and MPI_Send but in this functions the return from the function occurs immediately after the initialization of the receiving/transmitting process without waiting for the receipt/ processing of the entire message, so we can solve the problem with blocking operations in MPI_Send and MPI_Recv. In this lines the process waiting for their neareset neighbours and save information in int array buf and send information about yourself's rank to previous and next. The result is displayed on screens - ring topology works.

1.3 Appendix

The link to the sourse code which is placed on my github.