FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION

ITMO UNIVERSITY

Report

OpenMP. Practical tasks No. 1 – 3.
Parallel algorithms for the analysis and synthesis of data

Performed by
Aleksandr Shirokov
J4133c
Accepted by
Petr Andriushchenko

St. Petersburg
2021

Contents

1 Assignments **2**
1.1 Assignment 3. 2
1.1.1 Formulation of the problem 2
1.1.2 Example of launch parameters and output. Detailed description of solution . 2
1.2 Appendix 3

1 Assignments

1.1 Assignment 3.

1.1.1 Formulation of the problem

Compile and run ASSIGNMENT3.C program. Explain in detail how it works.

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

Code for **assignment 3** is here.
Compilation example: MPIC++ -O ./CPF/3.O ASSIGNMENT3.C
Launch example:

- Not in parallel: ./CPF/3.O

```
aptmess@improfeo: ~/ITMO/parallel_algorithms/HT/hw_mpi
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ mpic++ -o ./cpf/3.o Assignment3.c
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ ./cpf/3.o
Hello from process 0
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$
```

- In parallel: MPIRUN –OVERSUBSCRIBE -NP 4 ./CPF/3.O (have a problem without –OVERSUBSCRIBE option.

```
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ mpirun --oversubscribe -np 4 ./cpf/3.o
Hello from process 0
Hello from process 1
Hello from process 2
Hello from process 3
(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[]) {
    MPI_Init(&argc, &argv);
    int rank, n, i, message;
    MPI_Status status;
    MPI_Comm_size(MPI_COMM_WORLD, &n);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0)
    {
        cout << "Hello from process " << rank << "\n";
        for (i  = 1; i < n; i++) {
            MPI_Recv(&message, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
            cout << "Hello from process " << message << endl;
        }
    }
    else MPI_Send(&rank, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
    MPI_Finalize();
    return 0;
}

```

Assignment3 code

1. Line 5 - MPI_Init - initialisation, starting the parallel part with arguments of main function;
2. Line 6 - initialize variables, especially *rank* for rank of process and n for amount of process
3. Line 7 - creating MPI_status, variable status contains three attibutes of message:
   - MPI_source - number of the sending process;
   - MPI_tag - name of text message, identifier;
   - MPI_error - error code.
4. Line 7 - getting number of processes (n varialbe)
5. Line 8 - getting rank of the process (*rank* variable)
6. Line 10 - if statement
   - if rank == 0 - then we are printing 'Hello from process 0' and in range $i = [1..10]$ where i is number of process, waiting for incoming messages from any source (who is quicklier will be write earlier) with any tag using syntax of function MPI_Send. The information 'Hello from process %message' is printing after we get message on each iteration of cycle.
   - else - from processes with rank not 0 sending messages to process with rank 0. The message contains information about processes's rank.
7. Line 19 - Ending parallel part

I have analysed the first code with mpi, explain how it works, compile it and show results.

1.2 Appendix

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