FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION

ITMO UNIVERSITY

Report

MPI. Assignments 14 – 15

Parallel algorithms for the analysis and synthesis of data

Performed by
Aleksandr Shirokov
J4133c
Accepted by
Petr Andriushchenko
Deadline: 23.12.21

St. Petersburg
2021

Contents

1 Assignments

1.1 Assignment 14. MPI. Custom global operations. Custom global function.

1.1.1 Formulation of the problem

Understand the new functions in Assignment14.c.

Create your own global function for finding the maximum element, compare the correctness of execution with the MPI_MAX operation in the MPI_Reduce() function

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

Code for **assignment 14** is here.

Compilation example: mpic++ -o ./cpf/14.o Assignment14.c

Launch example: mpirun –oversubscribe -np 7 ./cpf/14.o

```
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ mpirun --oversubscribe -np 7 ./cpf/14.o
process 1 a[0] = 2
process 2 a[0] = 3
process 4 a[0] = 5
process 3 a[0] = 4
process 5 a[0] = 6
process 0 a[0] = 1
process 6 a[0] = 7
b[0] = 3
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$
```

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

```c
#include <stdio.h>
#include "mpi.h"
#define n 1000

void smod5(void *a, void *b, int *l, MPI_Datatype *type) {
    int i;
    for (i = 0; i < *l; i++)
        ((int*)b)[i] = (((int*)a)[i] + ((int*)b)[i]) % 5;
}

int main(int argc, char **argv)
{
    int rank, size, i;
    int a[n];
    int b[n];
    MPI_Init(&argc, &argv);
    MPI_Op op;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    for (i = 0; i < n; i++) a[i] = i + rank + 1;
    printf("process %d a[0] = %d\n", rank, a[0]);
    MPI_Op_create(&smod5, 1, &op);
    MPI_Reduce(a, b, n, MPI_INT, op , 0, MPI_COMM_WORLD);
    MPI_Op_free(&op);
    if (rank == 0) printf("b[0] = %d\n", b[0]);
    MPI_Finalize();
}
```

Assignment14 code

Firstly let's explain how this work code and after that will write the second part of the task. The idea of this program is to show hot to implement user-defined global function for some operations from class MPI_Op. For each process there is an initialization of array $a$by formula $a[i] = i + rank + 1, i = [1..n]$ where RANK is rank of the process and n is amount of processes. For example for 7 processes $a[0]$ for all processes is [1..7]. After that with operation MPI_Op_create which allows to create MPI_Op operation that can be used in reduction operations from an user-defined function and with second parameter 1 indicatesthat user-defined functon passed is communicative - in our example this function's name is SMOD5 (lines $5 - 9$). This user-defined function should have such parameters as:

- void **inputBuffer** - A pointer on the buffer providing the inputs of an MPI process. In our example the variable name is a;
- void **outputBuffer** - A pointer on the buffer in which write the reduction results. In our example the variable name is b;
- int **len** - The number of elements on which the reduction applies, amount of processes. In our example the variable name is l;
- MPI_Datatype* datatype - the datatype of output function

Our function SMOD5 implemented a logic - return an residual of sum of elements divided by 5. After that in line 23 we are using this written function as a reduce function and in the next line function MPI_Op_free deallocates an operation handle created with MPI_Op_create. As an output in line 25 we are showing result, which is storted in array b. Check if result if correct: $(1 + \ldots + 7)\%5 = 28\%5 = 3$ - as our program shows.

Let's implement a function for finding a maximum.

Code for **assignment 14 with maximum function** is here.

Compilation example: MPIC++ -O ./CPF/14.1.O ASSIGNMENT14.1.C

Launch example: MPIRUN –OVERSUBSCRIBE -NP 7 ./CPF/14.1.O

```
aptmess@improfeo: ~/ITMO/parallel_algorithms/HT/hw_mpi
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ mpic++ -o ./cpf/14.1.o Assignment14.1.c
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ mpirun --oversubscribe -np 7 ./cpf/14.1.o
process 0 a[0] = 1
process 1 a[0] = 2
process 6 a[0] = 7
process 5 a[0] = 6
process 2 a[0] = 3
process 4 a[0] = 5
process 3 a[0] = 4
b[0] = 7
b_check[0] = 7
Result of functions maximum and MPI_MAX are correct
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$
```

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

```cpp
#include <stdio.h>
#include "mpi.h"
#include <iostream>
#define n 1000

using namespace std;

void maximum(void *inputBuffer, void *outputBuffer, int *len, MPI_Datatype *type) {
    int* input = (int*)inputBuffer;
    int* output = (int*)outputBuffer;
    for (int i = 0; i < *len; i++)
        if (input[i] >= output[i]) output[i] = input[i];
}

int main(int argc, char **argv)
{
    int rank, size, i;
    int a[n];
    int b[n];
    int b_check[n];
    MPI_Init(&argc, &argv);
    MPI_Op op;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    for (i = 0; i < n; i++) a[i] = i + rank + 1;
    printf("process %d a[0] = %d\n", rank, a[0]);
    MPI_Op_create(&maximum, 1, &op);
    MPI_Reduce(a, b, n, MPI_INT, op , 0, MPI_COMM_WORLD);
    MPI_Op_free(&op);
    MPI_Reduce(a, b_check, n, MPI_INT, MPI_MAX , 0, MPI_COMM_WORLD);
    if (rank == 0)
    {
        printf("b[0] = %d\n", b[0]);
        printf("b_check[0] = %d\n", b[0]);
        int temp = 0;
        for (int i = 0; i < n; i++)
        {
            if (b[i] != b_check[i])
            {
                cout << "Wrong results of functions maximum and MPI_MAX" << endl;
                break;
            }
            else
            {
                temp += 1;
            }
        }
        if (temp == n)
        {
            cout << "Result of functions maximum and MPI_MAX are correct" << endl;
        }

    }
    MPI_Finalize();
}
```

Assignment14 code

I have implemented function maximum with simple logic for finding maximum (lines $8-13$). At lines $31-53$ i have implemented checking that results of my function is the same as using defined function MPI_MAX - if every piece of arrays b is equal to appropriate value b_check then the message that results are correct is displayed, else - that results isn't correct, but it is not our way - our results are correct! The program works correctly.

1.2 Assignment 15. MPI. Operations with communicators. Partitioning the communicator.

1.2.1 Formulation of the problem

Understand the new functions in Assignment15.c.
Append part of code.

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

Code for **assignment 15** is here.
Compilation example: mpic++ -o ./cpf/15.o Assignment15.c
Launch example: mpirun –oversubscribe -np 5 ./cpf/15.o
Let's move to the the code and explain how it works.

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

using namespace std;

int main(int argc, char **argv)
{
    int rank, size, i, rbuf;
    MPI_Init(&argc, &argv);
    MPI_Group group, new_group;
    MPI_Comm new_comm;
    int ranks[128], new_rank;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_group(MPI_COMM_WORLD, &group);

    for (i = 0; i < size / 2; i++) ranks[i] = i;

    if (rank < size / 2) MPI_Group_incl(group, size / 2, ranks, &new_group);
    else MPI_Group_excl(group, size / 2, ranks, &new_group);

    MPI_Comm_create(MPI_COMM_WORLD, new_group, &new_comm);
    MPI_Allreduce(&rank, &rbuf, 1, MPI_INT, MPI_SUM, new_comm);
    MPI_Group_rank(new_group, &new_rank);

    //Display values: "rank =, newrank =, rbuf ="

    cout << "process old rank=" << rank ;
    cout << ", newrank=" << new_rank;
    cout << ", rbuf=" << rbuf;
    cout << '\n' << endl;

    MPI_Finalize();
}
```

Assignment15 code

There are some new function in this code. First is MPI_Comm_group which accesses the group associated with given communicator, after that there are initialization of array ranks only for $i = \frac{size}{2}$ where $size$ is amount of processes. After that there are functions:

1. MPI_Group_incl - (
    - IN - MPI_Group **group** - group
    - IN - int **n** - number of elements in array ranks (and size of newgroup) (integer)

- IN - const int **ranks**[] - ranks of processes in group to appear in newgroup (array of integers)
- OUT - MPI_Group * **newgroup** - new group derived from above, in the order defined by ranks

)

2. MPI_Group_excl - Produces a group by reordering an existing group and taking only unlisted members (

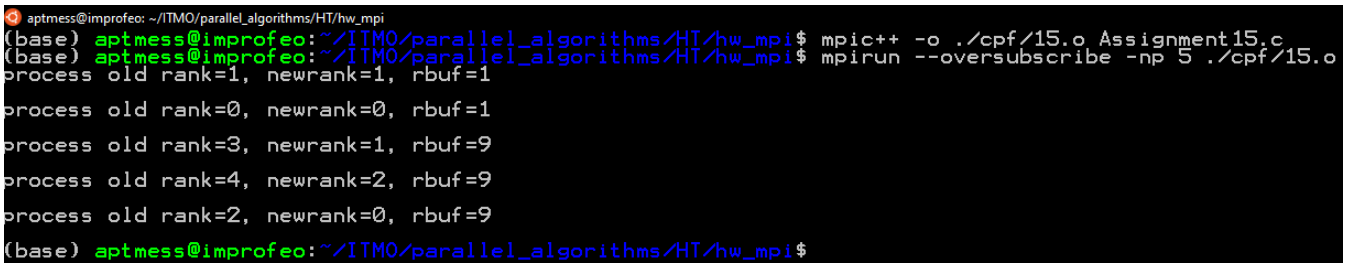
   - IN - MPI_Group **group** - group
   - IN - int **n** - number of elements in array ranks (integer)
   - IN - const int **ranks**[] - array of integer ranks in group not to appear in newgroup
   - OUT - MPI_Group * **newgroup** - new group derived from above, preserving the order defined by group

)

In our code it means that all processes with this functions are splitted into two groups depend on their's rank - less than $\frac{size}{2}$ goes to the new_group, more - to the second existing group. For new_group the new communcator with function MPI_Comm_create is creating and after that using function MPI_Allreduce which combines values from all processes and distributes the result back to all processes - returns sum of rank for new_group and this rank is written in variable new_rank as a new rank using function MPI_Group_rank. As a results we are displaying information abour rank if process, new rank of process and rbuf as a results of function MPI_Allreduce.

Let's see an example. Let amount of process be 5. Then processes of ranks $0, 1$ goes to group new_group and others are in group. rbuf is a sum of ranks for each group, so for first group it would be rbuf $= 0 + 1 = 1$ and for new_group the result is rbuf $= 2 + 3 + 4 = 9$. new_rank is new ranks for group new_group, so the processes with previous ranks $3, 4, 5$ maps appropriate to $0, 1, 2$ in new_group. For first group ranks are the same.

The results of example are on picture below



Results

The program works correctly!

1.3 Appendix

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