

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Параллельный алгоритмы»
Тема: Коллективные операции.

Студент гр. 9383

Ноздрин В.Я.

Преподаватель

Татаринов Ю.С.

Санкт-Петербург

2021

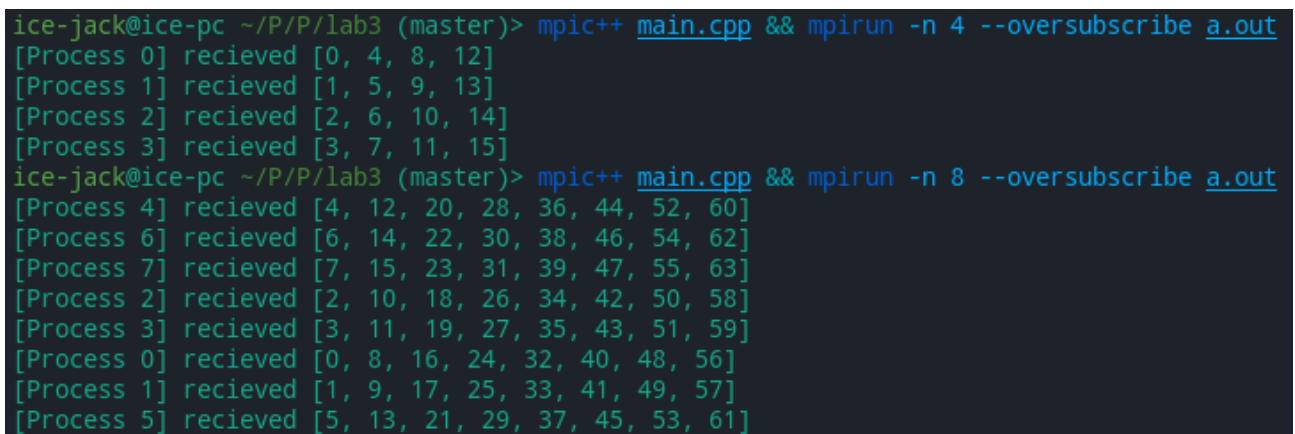
Задание. Вариант 5. Пляжный волейбол.

В каждом процессе дан набор из K чисел, где K — количество процессов. Используя функцию `MPI_Alltoall`, переслать в каждый процесс по одному числу из всех наборов: в процесс 0 — первые числа из наборов, в процесс 1 — вторые числа, и т. д. В каждом процессе вывести числа в порядке возрастания рангов переславших их процессов (включая число, полученное из этого же процесса).

Выполнение работы.

Была написана программа в которой каждый процесс генерирует сообщение в виде набора чисел по следующему правилу: процесс j генерирует сообщение с числами от $N*j$ до $(N+1)*j$, где N - количество процессов. Например, при 4 процессах процесс 0 будет генерировать сообщение $[0,1,2,3]$, процесс 1 будет генерировать сообщение $[4,5,6,7]$, и процессы 2 и 3 будут генерировать сообщения $[8,9,10,11]$ и $[12,13,14,15]$ соответственно.

Функция `MPI_Alltoall` отправляет сообщения от всех процессов ко всем процессом (для заданного коммутатора). При чем если рассмотрим процесс 0, то он отправит сообщения 0, 1, 2, 3 процессам с рангами 0, 1, 2 и 3 соответственно. При чем эти значение будут записаны в индекс 0 буфера для полученного сообщения. Процесс с рангом 0 получит сообщения 0, 4, 8, 12 от процессов 0, 1, 2, 3 соответственно. Ниже приведены скриншоты работы программы.



```
ice-jack@ice-pc ~/P/P/lab3 (master)> mpic++ main.cpp && mpirun -n 4 --oversubscribe a.out
[Process 0] recieved [0, 4, 8, 12]
[Process 1] recieved [1, 5, 9, 13]
[Process 2] recieved [2, 6, 10, 14]
[Process 3] recieved [3, 7, 11, 15]
ice-jack@ice-pc ~/P/P/lab3 (master)> mpic++ main.cpp && mpirun -n 8 --oversubscribe a.out
[Process 4] recieved [4, 12, 20, 28, 36, 44, 52, 60]
[Process 6] recieved [6, 14, 22, 30, 38, 46, 54, 62]
[Process 7] recieved [7, 15, 23, 31, 39, 47, 55, 63]
[Process 2] recieved [2, 10, 18, 26, 34, 42, 50, 58]
[Process 3] recieved [3, 11, 19, 27, 35, 43, 51, 59]
[Process 0] recieved [0, 8, 16, 24, 32, 40, 48, 56]
[Process 1] recieved [1, 9, 17, 25, 33, 41, 49, 57]
[Process 5] recieved [5, 13, 21, 29, 37, 45, 53, 61]
```

Рисунок 1. Запуск программы на 4 и на 8 процессах.

```

ice-jack@ice-pc ~/P/P/lab3 (master)> mpic++ main.cpp && mpirun -n 2 --oversubscribe a.out
[Process 0] recieved [0, 2]
[Process 1] recieved [1, 3]
ice-jack@ice-pc ~/P/P/lab3 (master)> mpic++ main.cpp && mpirun -n 3 --oversubscribe a.out
[Process 1] recieved [1, 4, 7]
[Process 2] recieved [2, 5, 8]
[Process 0] recieved [0, 3, 6]
ice-jack@ice-pc ~/P/P/lab3 (master)> mpic++ main.cpp && mpirun -n 4 --oversubscribe a.out
[Process 0] recieved [0, 4, 8, 12]
[Process 1] recieved [1, 5, 9, 13]
[Process 2] recieved [2, 6, 10, 14]
[Process 3] recieved [3, 7, 11, 15]
ice-jack@ice-pc ~/P/P/lab3 (master)> mpic++ main.cpp && mpirun -n 5 --oversubscribe a.out
[Process 2] recieved [2, 7, 12, 17, 22]
[Process 3] recieved [3, 8, 13, 18, 23]
[Process 0] recieved [0, 5, 10, 15, 20]
[Process 1] recieved [1, 6, 11, 16, 21]
[Process 4] recieved [4, 9, 14, 19, 24]

```

Рисунок 2. Запуск программы на 2, 3, 4 и 5 процессах.

При запуске программы на одном процессе, программа выводит сообщение на английском языке, сообщающее, что данная программа спроектирована для работы на как минимум двух процессах.

```

ice-jack@ice-pc ~/P/P/lab3 (master) [1]> mpic++ main.cpp && mpirun -n 1 a.out
This application is meant to be run with at least 2 MPI processes.
-----
MPI_ABORT was invoked on rank 0 in communicator MPI_COMM_WORLD
with errorcode 1.

NOTE: invoking MPI_ABORT causes Open MPI to kill all MPI processes.
You may or may not see output from other processes, depending on
exactly when Open MPI kills them.
-----

```

Рисунок 3. Запуск программы на одном процессе.

Выводы.

Получен опыт работы с коллективными операциями, изучена функция MPI_Alltoall.

ПРИЛОЖЕНИЕ

Файл main.cpp

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

int main(int argc, char* argv[]){
    int size, rank;
    MPI_Init(&argc, &argv);
    MPI_Status Status;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (size < 2) {
        std::cout << "This application is meant to be run with at
least 2 MPI processes.\n";
        MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
    }
    int sendbuf[size], recvbuf[size];
    for (int i = 0; i < size; i++) {
        sendbuf[i] = rank * size + i;
    }
    MPI_Alltoall(&sendbuf, 1, MPI_INT, recvbuf, 1, MPI_INT,
MPI_COMM_WORLD);
    std::cout << "[Process " << rank << "] recieved [" <<
recvbuf[0];
    for (int i = 1; i < size; i++)
        std::cout << ", " << recvbuf[i];
    std::cout << "]\n";
    MPI_Finalize();
    return 0;
}
```