**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Параллельные алгоритмы»**

ТЕМА: **Коллективные операции.**

|  |  |  |
| --- | --- | --- |
| Студент |  | Степаненко Д. В. |
| Преподаватель |  | Татаринов Ю. С. |

Санкт-Петербург

2023 г.

## Цель

Ознакомиться с коллективными операциями в библиотеке MPI. Написать программу с их использованием функции MPI\_Allgather.

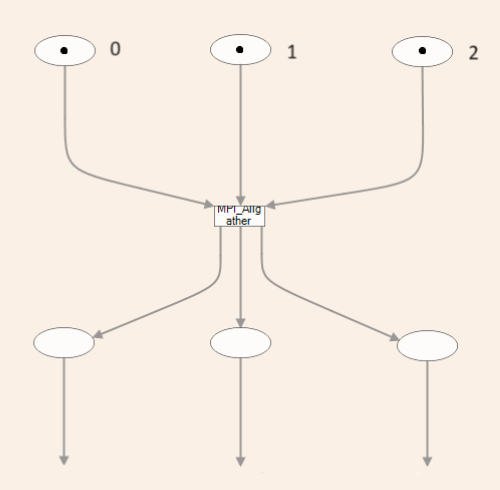
## Постановка задачи (вариант 4)

В каждом процессе даны четыре целых числа. Используя функцию MPI\_Allgather, переслать эти числа во все процессы и вывести их в каждом процессе в порядке возрастания рангов переславших их процессов (включая числа, полученные из этого же процесса).

## Выполнение работы

Программа создает несколько процессов, считывает ранг каждого и общее количество процессов. Далее в каждом процессе генерируются 4 целых числа. Они записываются в буфер отправки, пятым числом записывается ранг процесса. Стоит отметить, что генерируемые числа не будут совпадать с рангами процессов (это делается для удобного вывода). Далее происходит отправка и получение чисел со всех процессов с помощью функции MPI\_Allgather(). Таким образом, после выполнения функции у каждого процесса будет доступ ко всем числам, сгенерированным каждым процессом. Функция блокирует все процессы до момента получения и отправки данных каждому. Следующий шаг – вывод полученных чисел во всех процессах в порядке возрастания рангов. После завершается параллельная часть программы и освобождаются ресурсы.

Сеть Петри основной части алгоритма для трех процессов:



Листинг программы:

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

#include <time.h>

void generate\_numbers(int\* info, int ProcRank, int procNum){

for (int i = 0; i< 4; i++){

int new\_num = rand() %100 + 100 + ProcRank;

info[i] = new\_num;

}

info[4] = ProcRank;

}

int main(int argc, char\* argv[]) {

int procNum, ProcRank;

int send\_buf[5];

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &procNum);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcRank);

generate\_numbers(send\_buf,ProcRank,procNum);

int recv\_buf[5 \* procNum];

MPI\_Allgather(send\_buf, 5, MPI\_INT, recv\_buf, 5, MPI\_INT, MPI\_COMM\_WORLD);

//MPI\_Barrier(MPI\_COMM\_WORLD);

printf("Process %d recive:\n", ProcRank);

for (int i = 0; i < procNum; i++){

for(int j = 0; j < sizeof(recv\_buf)/sizeof(int); j++){

if(recv\_buf[j]==i){

printf("from process %d: %d, %d, %d, %d\n", i, recv\_buf[j-4], recv\_buf[j-3], recv\_buf[j-2], recv\_buf[j-1]);

}

}

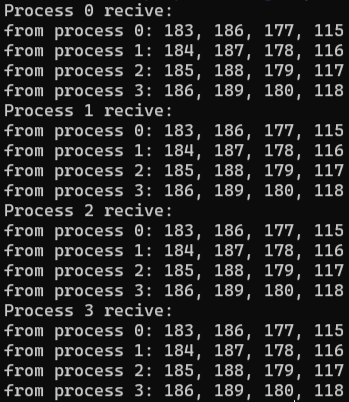
}

MPI\_Finalize();

return 0;

}

Полученный вывод при запуске на четырех процессах:



|  |  |
| --- | --- |
| Количество процессов (шт) | Среднее затрачиваемое время (мс) |
| 1 | 0,0062 |
| 2 | 0,0128 |
| 4 | 0,0234 |
| 6 | 0,0626 |
| 8 | 0,0958 |
| 10 | 0,0918 |
| 11 | 0,2319 |
| 12 | 0,41 |

Табл. 1 – Результаты работы программы на разном количестве процессов.

Расчеты ускорения программы выполним по формуле:

|  |  |
| --- | --- |
| Количество процессов P (шт) | Ускорение |
| 1 | 1 |
| 2 | 0,484 |
| 4 | 0,265 |
| 6 | 0,099 |
| 8 | 0,065 |
| 10 | 0,067 |
| 11 | 0,026 |
| 12 | 0,015 |

Табл. 2 – Результаты расчетов ускорения программы.

## Вывод

В ходе выполнения лабораторной работы были изучены коллективные операции в библиотеке MPI, использована на практике функция *MPI\_Allgather()*. Она пересылает данные другим процессам и агрегирует от остальных. С ее использованием была написана программа, удовлетворяющая ТЗ.

Время выполнения программы засекалось на двух функции: *MPI\_Allgather()*. Количество процессов изменялось от 1 до 12, т.к. дальнейшее увеличение будет малоинформативно. Благодаря параллельной отправке и сбору данных на каждом процессе, мы видим картину, что время выполнения программы увеличивается плавно. Увеличение происходит из-за коммуникационной задержки: при увеличении числа процессов, растет количество передаваемых данных (+5 за +1 процесс) и соединений, что приводит к возрастанию времени передачи данных.

Исходя из времени выполнения программы, можно сделать вывод, что и ускорение будет уменьшаться с увеличением процессов.