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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Параллельные алгоритмы»

Тема: Виртуальные топологии

Студент гр. 8303 _____ Почаев Н.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург

2020

Задание.

Вариант 8 **-** Количество процессов *K* равно 8 или 12, в каждом процессе дано вещественное число. Определить для всех процессов декартову топологию в виде трехмерной решетки размера *2×2 × K/4* (порядок нумерации процессов оставить прежним). Интерпретируя полученную решетку как *K/4* матриц размера *2×2* (в одну матрицу входят процессы с одинаковой третьей координатой), расщепить эту решетку на *K/4* указанных матриц. Используя одну коллективную операцию редукции, для каждой из полученных матриц *найти сумму исходных чисел и вывести найденные суммы в каждом процессе соответствующей матрицы*.

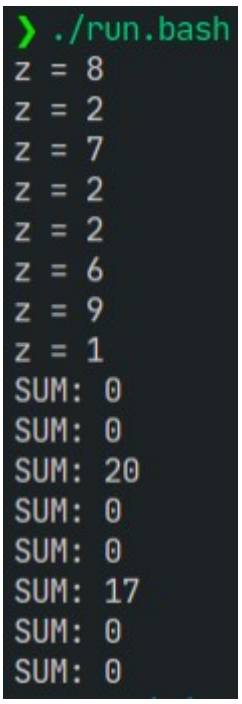
Описание решения.

Функция MPI_CART_CREATE возвращает дескриптор нового коммуникатора, к которому подключается топологическая информация. Если reorder = false, то номер каждого процесса в новой группе идентичен номеру в старой группе. В противном случае функция может переупорядочивать процессы (возможно, чтобы обеспечить хорошее наложение виртуальной топологии на физическую систему). Если полная размерность декартовой решетки меньше, чем размер группы коммуникаторов, то некоторые процессы возвращаются с результатом MPI_COMM_NULL по аналогии с MPI_COMM_SPLIT. Вызов будет неверным, если он задает решетку большего размера, чем размер группы.

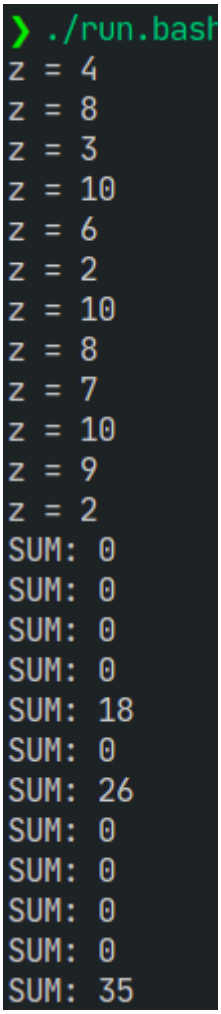
MPI_Cart_sub - функция выделения подпространства в декартовой топологии.

Функция MPI_Gather производит сборку блоков данных, посылаемых всеми процессами группы, в один массив процесса с номером root. Длина блоков предполагается одинаковой. Объединение происходит в порядке увеличения номеров процессов-отправителей. То есть данные, посланные процессом i из своего буфера sendbuf, помещаются в i-ю порцию буфера recvbuf процесса root. Длина массива, в который собираются данные, должна быть достаточной для их размещения.

Результат для 8 процессов представлен на скриншоте ниже:



Результат для 12 процессов представлен на скриншоте ниже:



Выводы.

В результате выполнения данной лабораторной работы была изучена работа с декартовой топологией в контексте библиотеки MPI.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
#include "mpi.h"
#include <iostream>
#include <cctype>
#include <random>
#include <vector>

// for random
using u32 = uint_least32_t;
using engine = std::mt19937;

int main(int ac,char **av)
{
    MPI_Init(&ac, &av);

    int rank, size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    // random generation
    std::random_device os_seed;
    const u32 seed = os_seed();
    engine generator(seed);
    std::uniform_int_distribution< u32 > distribute(1, 10);

    MPI_Comm nc, nc_sub;
     int res[4];
     int z = distribute(generator);
    std::cout << "z = " << z << std::endl;
     int dims[] = { 2, 2, size / 4 },
          periods[] = { 0, 0, 0 },
          remain_dims[] = { 1, 1, 0 };

    MPI_Cart_create(MPI_COMM_WORLD, 3, dims, periods, 0, &nc);
     MPI_Cart_sub(nc, remain_dims, &nc_sub);
     MPI_Gather(&z, 1, MPI_INT, res, 1, MPI_INT, 0, nc_sub);

    int sum = 0;
    for (auto &i : res) {
        sum += i;
    }
    std::cout << "SUM: " << sum << std::endl;

    MPI_Finalize();

    return 0;
}
```