**Практическая работа №1**

**Вариант №5**

**Формулировка задания.**

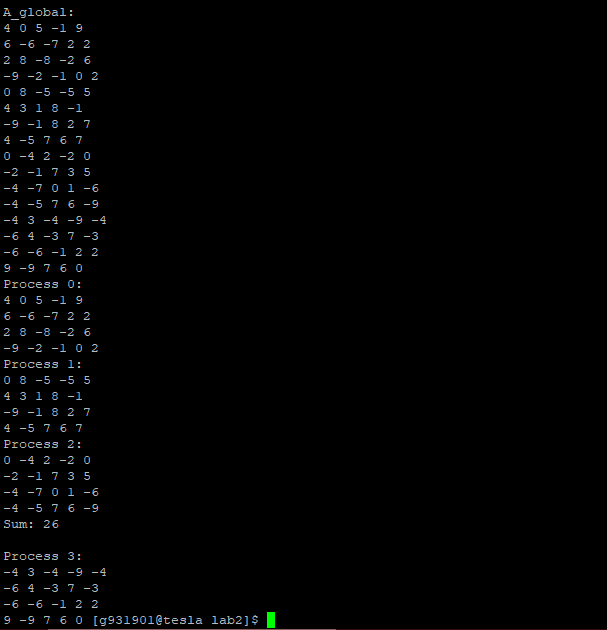
На процессе номер ноль заполнить матрицу A\_global[n][m] случайными числами из диапазона от -9 до 9. Размер массива задать равным n = 4\*size, m = 5, где size число используемых процессов. Распределить матрицу A\_global[][] по процессам. Вычислить сумму всех элементов массива. Выдать результат на процессе с номером 0

**Ход работы.**

Схема вычислений:

Сначала мы формируем матрицу с случайными числами на процессе ноль. Потом мы посылаем каждому процессу 4 строки в локальную матрицу с размерность (4 x m) и каждый раз смещаясь на 4 элемента в зависимости от процесса. После чего мы на каждом процессе считаем сумму локальной матрицы и отправляем на нулевом процессе результаты вычислений других процессов и выводим на экран.

Пример(size = 4, 12, 16):



****

****

**Заключение**

В данной работе были изучены коллективные операции в MPI. Была написана программа для распределения массива по процессам, подсчета суммы чисел в каждом локальном массиве, а затем сбора результата на нулевом процессе, с последующим выводом на экран.

**Код программы**

#include <stdio.h>

#include <mpi.h>

#include <cstdlib>

#include <ctime>

int main(int argc, char \*\*argv)

{

int rank, size, rc;

MPI\_Status status;

rc = MPI\_Init(&argc, &argv);

rc = MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

rc = MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

const int m = 5;

const int n = 4\*size;

int A\_global[n][m];

if(rank == 0) {

srand(time(nullptr));

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

for(int j = 0; j < m; j++) {

A\_global[i][j] = (rand() % 19) - 9;

}

}

printf("\nA\_global: ");

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

printf("\n");

for(int j = 0; j < m; j++) {

printf("%d ", A\_global[i][j]);

}

}

}

int A\_local[4][m];

int sendcounts[size];

int displs[size];

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

sendcounts[i] = m;

displs[i] = i\*m\*4;

}

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

MPI\_Scatterv(A\_global[i], sendcounts, displs, MPI\_INT, A\_local[i], m, MPI\_INT, 0, MPI\_COMM\_WORLD);

}

int sum\_local = 0;

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

for(int j = 0; j < m; j++) {

sum\_local += A\_local[i][j];

}

}

printf("\nProcess %d: ", rank);

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

printf("\n");

for(int j = 0; j < m; j++) {

printf("%d ", A\_local[i][j]);

}

}

int sum\_global = 0;

MPI\_Reduce(&sum\_local, &sum\_global, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (rank == 0) {

printf("\nSum: %d\n", sum\_global);

}

rc = MPI\_Finalize();

}

**Ссылка на код на кластере**

/home/g931901/g932204/Sobol/lab2