Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №4

з дисципліни: “ Паралельні та розподілені обчислення”

на тему:

“ **ПАРАЛЕЛЬНІ АЛГОРИТМИ МНОЖЕННЯ МАТРИЦІ НА ВЕКТОР.**”

Виконав: ст. гр. КІ-34

Гриценко В.В.

Прийняв: викл.

Козак Н.Б.

Львів

2020

**Мета роботи:** Ознайомитись з методами організації паралельного множення матриці на вектор та розробити паралельну програму з використанням технології MPI.

**Завдання:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ варіанту** | **Розмір матриці** | | **Тип розбиття** | **Кількість процесорів** |
| 3 | 150 | 570 | стрічкове(гор) | 5 |

**Виконання роботи:**

При горизонтальному способі розбиття даних (розбиття матриці на горизонтальні смуги) вхідна матриця буде мати такий вигляд:

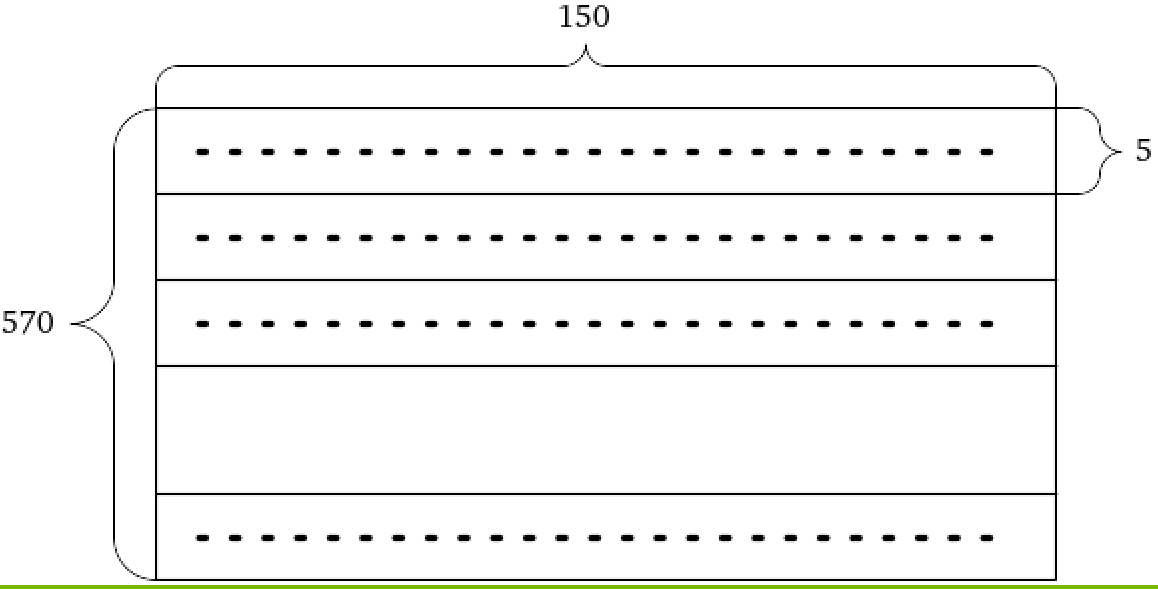


Рис. 1. Розбиття вхідної матриці на горизонтальні смуги для 5 процесорів.

Для кожного процесора визначено наступний розмір блоку для таких параметрів: матриця *А* розмірності *m х n (570 x 150)*, вектор *b*, що складається з *n (150)* елементів та вектора результатів *с* розміру *m (150).* Вважається, що вектори *b* і *c* копіюються на кожний процесор.

Тоді: m x n / p + n + m = *570 x 150*/ 5 + 150 + *570* = 17820 елементів;

Кількість операцій визначається на основі *T1 = m·(2n-1)* та становить 170850 операцій для кожного процесора.

Як базова підзадача може бути вибрана операція скалярного множення одного рядка матриці на вектор.

**Текст програми:**

#include <fstream>

#include <iomanip>

#include "mpi/mpi.h"

using namespace std;

// Данні

const int TAG = 0,

ROOT = 0,

PROCESSORS\_COUNT = 5,

MATRIX\_X = 150,

MATRIX\_Y = 570;

int main()

{

int rank, commSize;

double \*\* A = new double\*[MATRIX\_Y],

\* b = new double[MATRIX\_X],

\* c = new double[MATRIX\_Y];

// A, b initialization

// A[570][150]

for(int i = 0; i < MATRIX\_Y; ++i){

A[i] = new double[MATRIX\_X];

for(int j = 0; j < MATRIX\_X; ++j){

A[i][j] = rand() % 10; // 0..9

}

}

// b[150]

for(int i = 0; i < MATRIX\_X; ++i){

// b

b[i] = rand() % 10; // 0..9

}

// RESULT

double\* res = new double[MATRIX\_Y];

MPI\_Init(nullptr, nullptr);

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

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

if(commSize < PROCESSORS\_COUNT)

{

cerr << "MPI\_COMM\_WORLD has less commSize(" << commSize << ") than PROCESSORS\_COUNT." << endl;

return 1;

}

commSize = PROCESSORS\_COUNT;

// every process take it's row and save calculations on the right place at res[570] vector

for(int i = rank; i < MATRIX\_Y;){

for(int j = 0; j < MATRIX\_X; ++j)

res[i] += A[i][j] \* b[j];

i += commSize;

}

MPI\_Finalize();

ofstream out("result.txt", ios::trunc | ios::out);

out << "A matrix: ---------------------------------------------------------------------" << endl;

for(int i = 0; i < MATRIX\_Y; ++i)

{

for(int j = 0; j < MATRIX\_X; ++j)

out << setw(2) << A[i][j];

out << endl;

}

out << endl << "b vector: ---------------------------------------------------------------------" << endl;

for(int i = 0; i < MATRIX\_Y; ++i)

out << setw(2) << b[i];

out << endl << endl << "RESULT c vector: ---------------------------------------------------------------------" << endl;

for(int i = 0; i < MATRIX\_Y; ++i)

out << res[i] << " ";

return(0);

}

**Результат виконання програми:**

Результатом виконання програми є створення файлу result.txt з даними про матрицю «А», вектор «b» та вектор-результат «с» - результуючий вектор від множення «A\*b».

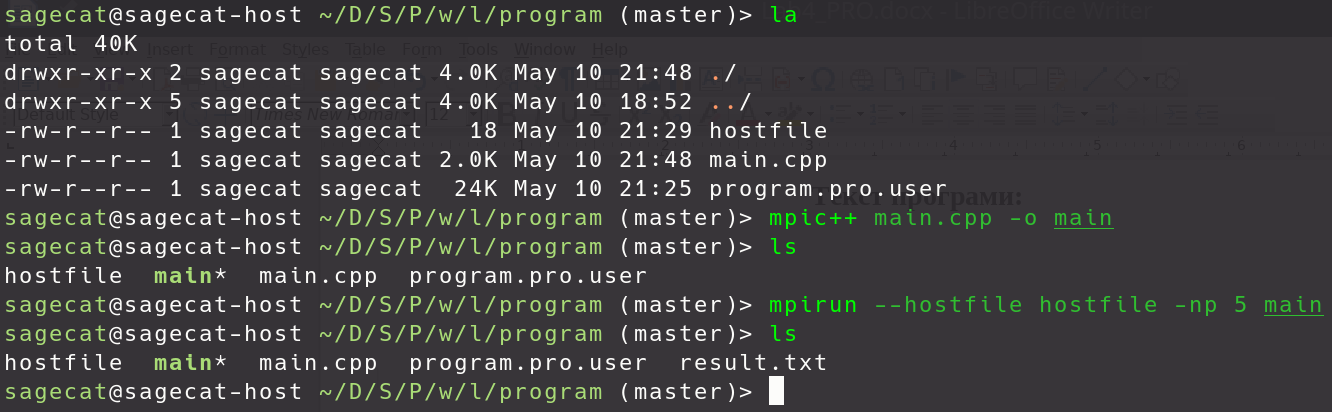
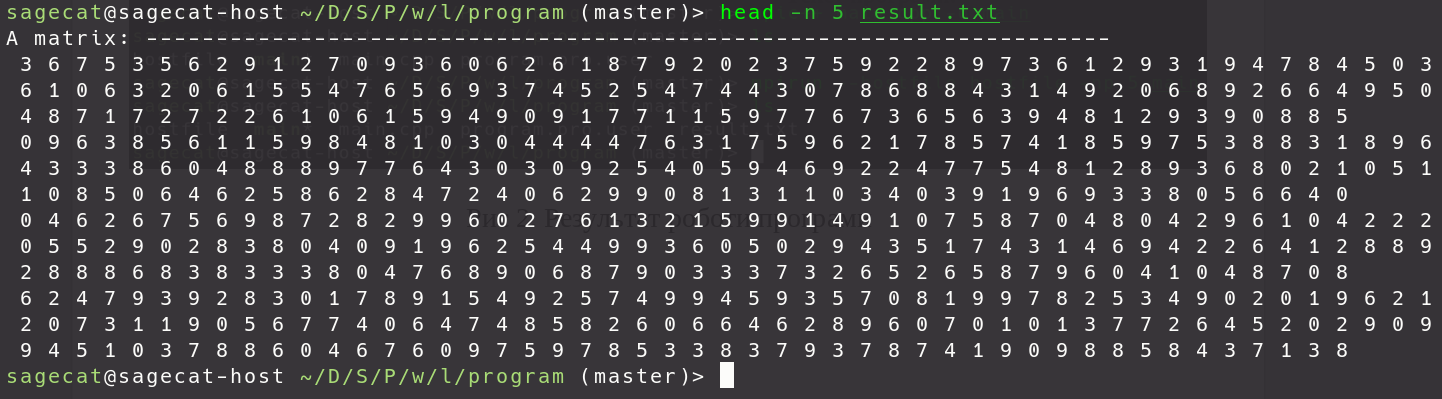
****

Рис 2. Компіляція і виконання програми

Рис 3. 5 верхніх рядків файлу result.txt

**Висновок:** на даній лабораторній роботі розроблено алгоритм паралельного перемноження матриці на вектор при стрічковому горизонтальному розбитті вхідних даних. Виконано його програмну реалізацію з використанням МРІ. Розроблено схему інформаційної взаємодії між підзадачами та виконано їх масштабування на задану кількість проесорів системи. Обчислено кількість елементів та операцій для кожного процесора.