Міністерство освіти і науки, молоді та спорту України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи № 1

з дисципліни: “Паралельні та розподілені обчислення”

на тему: “Використання функціональної декомпозиції для розв’язку обчислювальних задач”

Виконав: студент гр.. КІ-34

Юсько В.А.

Прийняв:

Козак Н. Б.

Львів

2020

**Мета роботи:** Вивчити методи декомпозицій задач. Набути навиків розв’язування задач з використанням функціональної декомпозиції

**Завдання:**

Використовуючи метод функціональної декомпозиції, розробити алгоритм обчислення запропонованого матрично-векторного виразу, який би враховував можливість паралельного виконання і був оптимальним з точки зору часових затрат.

На основі створеного алгоритму написати програму яка дозволяє обчислити вираз та ілюструє проведену декомпозицію

**Варіант завдання – 16**

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | число | | |
| bi=16/(i3) | A1(b1+16c1) | A2(B2+16C2)  Cij=16/(i+j)2 |

**Схема декомпозиції задачі**



**Текст програми**

// PRO\_lab1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include "pch.h"

#include <iostream>

#include "omp.h"

#include <ctime>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int size;

cout << " Введiть порядок n: ";

cin >> size;

int\*\* A = new int\*[size]; // а - указатель на указатели, создается массив указателей

cout << " \t \============\ \tЗаповнення матрицi A.. \t\t \============\ \n";

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

{

A[i] = new int[size];

for (int j = 0; j < size; j++)

{

A[i][j] = rand() % 50 + 1;

}

}

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

{

cout << "\t";

for (int j = 0; j < size; j++)

{

cout << "\t" << A[i][j];

}

cout << endl;

}

int\*\* A1 = new int\*[size];

cout << " \t \============\ \tЗаповнення матрицi A1.. \t \============\ \n";

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

{

A1[i] = new int[size];

for (int j = 0; j < size; j++)

{

A1[i][j] = rand() % 50 + 1;

}

}

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

{

cout << "\t";

for (int j = 0; j < size; j++)

{

cout << "\t" << A1[i][j];

}

cout << endl;

}

int \*b1 = new int[size];

cout << " \t \============\ \tЗаповнення вектора-стовпця b1.. \============\ \n";

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

{

b1[i] = rand() % 50 + 1;

}

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

{

cout << "\t \t " << b1[i] << endl;

}

int \*c1 = new int[size];

cout << " \t \============\ \tЗаповнення вектора-стовпця c1.. \============\ \n";

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

{

c1[i] = rand() % 50 + 1;

}

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

{

cout << "\t \t " << c1[i] << endl;

}

int\*\* A2 = new int\*[size];

cout << " \t \============\ \tЗаповнення матрицi A2.. \t \============\ \n";

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

{

A2[i] = new int[size];

for (int j = 0; j < size; j++)

{

A2[i][j] = rand() % 50 + 1;

}

}

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

{

cout << "\t";

for (int j = 0; j < size; j++)

{

cout << "\t" << A2[i][j];

}

cout << endl;

}

int\*\* B2 = new int\*[size];

cout << " \t \============\ \tЗаповнення матрицi B2.. \t \============\ \n";

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

{

B2[i] = new int[size];

for (int j = 0; j < size; j++)

{

B2[i][j] = rand() % 50 + 1;

}

}

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

{

cout << "\t";

for (int j = 0; j < size; j++)

{

cout << "\t" << B2[i][j];

}

cout << endl;

}

double \*b = new double[size];

int \*y1 = new int[size];

int \*y2 = new int[size];

double\*\* C2 = new double\*[size];

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

{

C2[i] = new double[size];

}

double\*\* Y3 = new double\*[size];

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

{

Y3[i] = new double[size];

}

double\*\* T1 = new double\*[size];

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

{

T1[i] = new double[size];

}

double\*\* Y3\_2 = new double\*[size];

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

{

Y3\_2[i] = new double[size];

}

int \*t2 = new int[size];

double \*t3 = new double[size];

double \*t4 = new double[size];

double \*t5 = new double[size];

double \*t6 = new double[size];

double \*t7 = new double[size];

double t8;

double t9;

double res;

// 1-й рівень

#pragma omp parallel sections shared(b, C2, size)

{

#pragma omp section for private(i)

{

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

{

b[i] = 16 / pow((i + 1), 3);

}

}

#pragma omp section for private(i, j)

{

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

{

for (int j = 0; j < size; j++)

{

C2[i][j] = 16 / (pow((i + j + 2), 2));

}

}

}

}

// 2-й рівень

#pragma omp parallel sections shared(b1, B2, c1, C2, T1, t2, size)

{

#pragma omp section for private(i, j)

{

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

{

T1[i] = new double[size];

for (int j = 0; j < size; j++)

{

T1[i][j] = B2[i][j] + 16 \* C2[i][j];

}

}

}

#pragma omp section for private(i)

{

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

{

t2[i] = b1[i] + 16 \* c1[i];

}

}

}

// 3-й рівень

#pragma omp parallel sections shared (Y3, A2, T1, y1, y2, size)

{

#pragma omp section for private(i, j, k)

{

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

{

for (int j = 0; j < size; j++)

{

Y3[i][j] = 0;

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

Y3[i][j] += (A2[i][k] \* T1[k][j]);

}

}

}

}

#pragma omp section for private(i, j)

{

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

{

y2[i] = 0;

for (int j = 0; j < size; j++)

{

y2[i] += A1[i][j] \* t2[j];

}

}

}

#pragma omp section for private(i. j)

{

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

{

y1[i] = 0;

for (int j = 0; j < size; j++)

{

y1[i] += A[i][j] \* b[j];

}

}

}

}

// 4-й рівень

#pragma omp parallel sections shared (t3, t4, t5, Y3, Y3\_2 y2, y1, size)

{

#pragma omp section for private(i, j)

{

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

{

t3[i] = 0;

for (int j = 0; j < size; j++)

{

t3[i] += Y3[i][j] \* y2[j];

}

}

}

#pragma omp section for private(i, j)

{

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

{

for (int j = 0; j < size; j++)

{

Y3\_2[i][j] = 0;

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

Y3\_2[i][j] += (Y3[i][k] \* Y3[k][j]);

}

}

}

}

#pragma omp section for private(i, j)

{

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

{

t5[i] = 0;

for (int j = 0; j < size; j++)

{

t5[i] += Y3[i][j] \* y1[j];

}

}

}

}

// 5-й рівень

#pragma omp parallel sections shared(size, t3, t4, t5, t6, t7, y2)

{

#pragma omp section for private(i, j)

{

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

{

t4[i] = 0;

for (int j = 0; j < size; j++)

{

t4[i] += Y3\_2[i][j] \* y2[j];

}

}

}

#pragma omp section for private(i)

{

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

{

t6[i] = t3[i] + y2[i];

}

}

}

// 6-й рівень

#pragma omp parallel sections shared(t6, t7, t8, t9, y1, y2, size)

{

#pragma omp section for private(i)

{

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

{

t7[i] = t4[i] + t5[i];

}

}

#pragma omp section private(i) reduction(+: t9)

{

t9 = 0;

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

{

t9 += y1[i] \* y2[i];

}

}

// 7-й рівень

#pragma omp private(i)reduction(+: t8)

{

t8 = 0;

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

{

t8 += t6[i] \* t7[i];

}

}

}

// 8-й рівень

#pragma omp single

{

res = t8 + t9;

}

cout << endl << " \t \============\ \tПаралельне обчислення.. \t \============\ \n";

cout << "\t \============\ \t\t" << res << "\t\t \============\ \n" << endl;

// перевірка

// 1-й рівень

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

{

for (int j = 0; j < size; j++)

{

C2[i][j] = 16 / (pow((i + j + 2), 2));

}

}

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

{

b[i] = 16 / pow((i + 1), 3);

}

// 2-й рівень

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

{

T1[i] = new double[size];

for (int j = 0; j < size; j++)

{

T1[i][j] = B2[i][j] + 16 \* C2[i][j];

}

}

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

{

t2[i] = b1[i] + 16 \* c1[i];

}

// 3-й рівень

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

{

for (int j = 0; j < size; j++)

{

Y3[i][j] = 0;

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

Y3[i][j] += (A2[i][k] \* T1[k][j]);

}

}

}

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

{

y2[i] = 0;

for (int j = 0; j < size; j++)

{

y2[i] += A1[i][j] \* t2[j];

}

}

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

{

y1[i] = 0;

for (int j = 0; j < size; j++)

{

y1[i] += A[i][j] \* b[j];

}

}

// 4-й рівень

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

{

t3[i] = 0;

for (int j = 0; j < size; j++)

{

t3[i] += Y3[i][j] \* y2[j];

}

}

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

{

for (int j = 0; j < size; j++)

{

Y3\_2[i][j] = 0;

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

Y3\_2[i][j] += (Y3[i][k] \* Y3[k][j]);

}

}

}

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

{

t5[i] = 0;

for (int j = 0; j < size; j++)

{

t5[i] += Y3[i][j] \* y1[j];

}

}

// 5-й рівень

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

{

t4[i] = 0;

for (int j = 0; j < size; j++)

{

t4[i] += Y3\_2[i][j] \* y2[j];

}

}

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

{

t6[i] = t3[i] + y2[i];

}

// 6-й рівень

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

{

t7[i] = t4[i] + t5[i];

}

t9 = 0;

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

{

t9 += y1[i] \* y2[i];

}

// 7-й рівень

t8 = 0;

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

{

t8 += t6[i] \* t7[i];

}

// 8-й рівень

res = t8 + t9;

cout << endl << " \t \============\ \tПеревiрка.. \t\t\t \============\ \n";

cout << "\t \============\ \t\t" << res << "\t\t \============\ \n" << endl;

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

delete[] A[i];

delete[] A1[i];

delete[] A2[i];

delete[] B2[i];

delete[] C2[i];

delete[] Y3[i];

delete[] Y3\_2[i];

delete[] T1[i];

}

delete[] A;

delete[] A1;

delete[] A2;

delete[] b;

delete[] b1;

delete[] B2;

delete[] c1;

delete[] C2;

delete[] Y3;

delete[] Y3\_2;

delete[] T1;

delete[] t2;

delete[] t3;

delete[] t4;

delete[] t5;

delete[] t6;

delete[] t7;

return 0;

}

**Результат роботи**

Введiть порядок n: 4

============ Заповнення матрицi A.. ============

7 48 41 15

44 38 43 24

40 49 16 41

40 44 27 16

============ Заповнення матрицi A1.. ============

32 2 34 24

8 37 46 18

1 2 35 14

34 44 7 8

============ Заповнення вектора-стовпця b1.. ============

50

26

5

24

============ Заповнення вектора-стовпця c1.. ============

28

31

41

32

============ Заповнення матрицi A2.. ============

24 25 44 28

6 24 29 16

38 30 21 17

37 42 47 33

============ Заповнення матрицi B2.. ============

46 10 11 7

41 48 13 6

2 1 37 16

3 14 3 11

============ Паралельне обчислення.. ============

============ 3.41559e+22 ============

============ Перевiрка.. ============

============ 3.41559e+22 ============

**Перевірка за допомогою MATLAB**

A = [7 48 41 15 ; 44 38 43 24 ; 40 49 16 41 ; 40 44 27 16];

A1 = [32 2 34 24 ; 8 37 46 18 ; 1 2 35 14 ; 34 44 7 8];

b1 = [50 ; 26 ; 5 ; 24];

c1 = [28 ; 31 ; 41 ; 32];

A2 = [24 25 44 28 ; 6 24 29 16 ; 38 30 21 17 ; 37 42 47 33];

B2 = [46 10 11 7 ; 41 48 13 6 ; 2 1 37 16 ; 3 14 3 11];

% level 1

for i = 1 : 4

for j = 1 : 4

C2(i, j) = 16 / (i + j)^2;

end

end

for i = 1 : 4

b(i) = 16 / (i)^3;

end

% level 2

%T1 = B2 + 16 \* C2;

%t2 = b1 + 16 \* c1;

% level 3

Y3 = A2 \* (B2 + 16 \* C2);

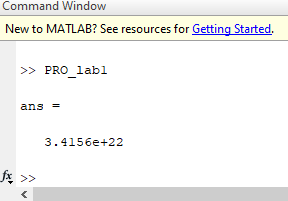
y2 = A1 \* (b1 + 16 \* c1);

y1 = A \* b.';

% level 8

(Y3 \* y2 + y2).' \* (Y3^2 \* y2 + Y3 \* y1) + y1.' \* y2

**Результат MATLAB**



**Висновок**

Під час виконання даної лабораторної роботи була виконана декомпозиція задачі, її програмна реалізація з використанням технології OpenMP та перевірка результату програми за допомогою MATLAB