Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №1

з дисципліни: “Паралельні та розподілені обчислення”

на тему: “ Функціональна декомпозиція на базі MPI ”

Виконав: ст. гр. КІ-33

Лобай Р.І

Прийняв: викладач

Козак Н.Б.

Львів – 2020

**Мета:** Вивчити методи декомпозиції задач. Набути навиків розв’язування задач з використанням функціональної декомпозиції. Освоїти інструменти бібліотеки MPI для реалізації декомпозиції задач.

**Теоретичні відомості**

**1.Використання функціональної декомпозиції**

Найважливішим та найважчим етапом при створенні програми є розробка алгоритму, особливо, якщо мова йде про паралельний алгоритм. Процес створення паралельного алгоритму можна розбити на чотири кроки.

**1.1. Декомпозиція.** На цьому етапі вихідна задача аналізується, оцінюється можливість її розпаралелювання. Іноді виграш від розпаралелення може бути незначним, а трудоємкість розробки паралельної програми велика. В цьому випадку перший крок розробки алгоритму виявляється і останнім. Якщо ж ситуація відмінна від описаної, то задача та пов’язані з нею дані розділяються на дрібніші частини – підзадачі і фрагменти структур даних. Особливості архітектури конкретної обчислювальної системи на цьому етапі можуть не враховуватися.

**1.2. Проектування комунікацій (обміну даними) між задачами.** На цьому етапі визначаються зв’язки, необхідні для пересилання вхідних даних, проміжних результатів виконання підзадач, а також комунікації, що необхідні для керування роботою під задач. Обираються методи та алгоритми комунікацій.

**1.3.Укрупнення.** Підзадачі можуть об’єднуватися у більші блоки, якщо це дозволяє підвищити ефективність алгоритму і знизити трудоємкість розробки. Основними критеріями на даному кроці є ефективність алгоритму (в першу чергу‑продуктивність) та трудоємкість його реалізації.

**1.4. Планування обчислень.** На цьому кроці виконується розподіл під задач між процесорами. Основний критерій вибору способу розміщення під задач – ефективне використання процесорів з мінімальними затратами часу на обмін даними.

**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 .

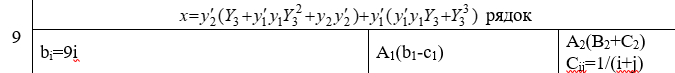
**Правила знаходження елементів виразу.**

1). Задати\* квадратну матрицю А порядку n. Отримати вектор(стовпець) , де b – вектор-стовпець, елементи якого обраховуються за формулою, згідно варіанту.

2). Задати квадратну матрицю А1 порядку n та вектори-рядки b1 та c1 з n елементами кожен. Отримати вектор- рядок  згідно формули, що задається варіантом.

3). Задати квадратні матриці А2 та B2 порядку n. Отримати матрицю , яка залежить від А2, B2 та додатково визначеної матриці С2, елементи якої знаходяться за формулою, вказаною варіантом.

**Варіант 9:**

****

**Декомпозиція для заданого виразу**



Рис.1 Схема декомпозиції обчислення виразу

**Результати роботи програми на MPI**

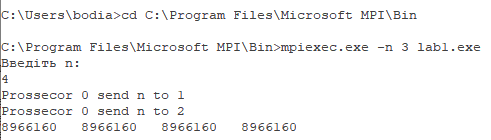


Рис.2. Паралельне виконання за допомого MPI

**Код програми:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

using MPI;

namespace lab1

{

static class Program

{

static int n = 0;

private static float[] y1;

private static float[] y2;

private static float[,] Y3;

static void Main(string[] args)

{

using (new MPI.Environment(ref args))

{

Intracommunicator comm = Communicator.world;

switch (comm.Rank)

{

case 0:

{

EnterN();

comm.Send(n, 1, 0);

Console.WriteLine("Prossecor 0 send n to 1");

comm.Send(n, 2, 0);

Console.WriteLine("Prossecor 0 send n to 2");

//1 етап

float[] b = initVector\_b();

float[,] A = fillDataInMatrix();

y1 = mulVectorToMatrix(b, A, true);

Y3 = comm.Receive<float[,]>(2,1);

//2 етап

float[,] y1Ty1 = mulColumnOnRow(y1,y1);

comm.Send<float[,]>(y1Ty1,2,2);

float[,] Y3Y3Y3 = comm.Receive<float[,]>(2,3);

//3 етап

float[,] Y3y1Ty1 = MatrixMulMatrix(Y3, y1Ty1);

//4 етап

float[,] y1Ty1Y3addY3Y3Y3 = calc\_Sum\_or\_Sub\_Matrix(Y3y1Ty1, Y3Y3Y3,true);

//5 етап

float[] res0 = mulVectorToMatrix(y1, y1Ty1Y3addY3Y3Y3,true);

comm.Send<float[]>(res0,2,6);

//6 етап

break;

}

case 1:

{

n = comm.Receive<int>(0, 0);

//1 етап

float[] b1 = fillDataInVector();

float[] c1 = fillDataInVector();

float[,] A1 = fillDataInMatrix();

float[] b1subc1 = calc\_Sum\_Or\_Sub\_Vector(b1,c1,false);

y2 = mulVectorToMatrix(b1subc1, A1, false);

comm.Send<float[]>(y2,2,7);

//2 етап

float[,] y2Ty2 = mulColumnOnRow(y2, y2);

comm.Send<float[,]>(y2Ty2, 2,4);

//3 етап

//--------------

//4 етап

//5 етап

break;

}

case 2:

{

n = comm.Receive<int>(0, 0);

////1 етап

float[,] C2 = initMatrix\_C2();

float[,] A2 = fillDataInMatrix();

float[,] B2 = fillDataInMatrix();

float[,] B2addC2 = calc\_Sum\_or\_Sub\_Matrix(B2, C2, false);

Y3 = MatrixMulMatrix(A2, B2addC2);

comm.Send<float[,]>(Y3,0,1);

// 2 етап

float[,] Y3Y3 = MatrixMulMatrix(Y3,Y3);

float[,] Y3Y3Y3 = MatrixMulMatrix(Y3Y3,Y3);

comm.Send<float[,]>(Y3Y3Y3,0,3);

float[,] y1Ty1 = comm.Receive<float[,]>(0, 2);

// 3 етап

float[,] Y3Y3y1Ty1 = MatrixMulMatrix(Y3Y3,y1Ty1);

float[,] y2Ty2 = comm.Receive<float[,]>(1,4);

//4 етап

float[,] Y3Y3y1Ty1addy2Ty2 = calc\_Sum\_or\_Sub\_Matrix(Y3Y3y1Ty1, y2Ty2,true);

// 5 етап

float[,] Y3Y3y1Ty1addy2Ty2addY3 = calc\_Sum\_or\_Sub\_Matrix(Y3Y3y1Ty1addy2Ty2,Y3,true);

// 6 етап

float[] y2 = comm.Receive<float[]>(1,7);

float[] res2 = mulVectorToMatrix(y2, Y3Y3y1Ty1addy2Ty2addY3,true);

float[] res0 = comm.Receive<float[]>(0, 6);

// 7 етап

float[] res = calc\_Sum\_Or\_Sub\_Vector(res0,res2,true);

printVector(res,true);

break;

}

}

}

}

static float[,] MatrixMulMatrix(float[,] A, float[,] B)

{

float[,] Result = new float[n, n];

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

{

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

{

Result[i, j] = 0;

for (int k = 0; k < n; k++)

Result[i, j] += A[i, k] \* B[k, j];

}

}

return Result;

}

static float[,] mulColumnOnRow(float[] a, float[] b)

{

float[,] res = new float[n, n];

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

{

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

{

res[i, j] += a[i] \* b[i];

}

}

return res;

}

static float[,] calc\_Sum\_or\_Sub\_Matrix(float[,] A, float[,] B, bool temp)

{

float[,] res = new float[n, n];

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

{

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

{

if (temp)

res[i, j] = A[i, j] + B[i, j];

else

res[i, j] = A[i, j] - B[i, j];

}

}

return res;

}

static float[] calc\_Sum\_Or\_Sub\_Vector(float[] A, float[] B, bool temp)

{

float[] res = new float[n];

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

{

if (temp)

res[i] = A[i] + B[i];

else

res[i] = A[i] - B[i];

}

return res;

}

static void printVector(float[] vec, bool b)

{

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

{

if (b)

{

Console.Write(vec[i] + " ");

}

else

{

Console.Write(vec[i] + " " + "\n");

}

}

Console.WriteLine();

}

static float[] mulVectorToMatrix(float[] vec, float[,] mat, bool b)

{

float[] res = new float[n];

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

{

res[i] = 0;

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

{

if (b)

{

//матриця на стовбець

res[i] += mat[i, j] \* vec[j];

}

else

{

//строка на матрицю

res[i] += vec[j] \* mat[j, i];

}

}

}

return res;

}

static void EnterN()

{

Console.WriteLine("Введiть n:");

string NText = Console.ReadLine();

while (!Int32.TryParse(NText, out n) && n > 3)

{

Console.WriteLine("Помилка! Введено неправильне n");

NText = Console.ReadLine();

}

}

static float[] initVector\_b()

{

float[] b = new float[n];

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

{

b[i] =9 \* i;

}

return b;

}

static float[,] fillDataInMatrix()

{

float[,] temp = new float[n, n];

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

{

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

{

temp[i, j] = 1;

}

}

return temp;

}

static float[] fillDataInVector()

{

float[] temp = new float[n];

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

{

temp[i] = 1;

}

return temp;

}

static float[,] initMatrix\_C2()

{

float[,] C2 = new float[n, n];

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

{

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

{

if (i == 0)

{

C2[i, j] = (float)(1 / (i + 1 + j));

}else

{

C2[i, j] = (float)(1 / (i + j));

}

}

}

return C2;

}

}

}

**Висновок:** під час лабораторної роботи я ознайомився з методами організації паралельного множення матриці на вектор та розробив паралельну програму з використанням технології MPI.