**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**



Звіт

до лабораторної роботи № 2

з дисципліни «Паралельні та розподілені обчислення»

на тему: «Паралельне представлення алгоритмів»

Виконав:

ст.гр. КІ-34

Мороз О.Ю.

Прийняв:

Козак Н.Б.

**Львів 2020**

Мета роботи: Вивчити можливості паралельного представлення алгоритмів. Набути навиків такого представлення.

**Завдання:**

Запропонувати та реалізувати локально-рекурсивний алгоритм обчислення виразу: , де *А* та *В* матриці з елементами  та , відповідно . Матриця *А* задається однозначно і залежить лише від розмірності даних. Для матриці *В*: заштрихована область — довільні цілі числа, відмінні від нуля, а незаштрихована область – нулі.

|  |  |  |
| --- | --- | --- |
| **2** | 1\*2 0 ... 0  0 2\*3 ... 0  ....  0 .... n(n+1) |  |

1. Програма з одноразовим присвоюванням.

Програма об’єднана з програмою реалізації оптимізованого локально-рекурсивного алгоритму, і подана в пункті 6.

2. Рекурсивні рівняння: *Cij(k+1)=Cij(k)+Aij(k)\*Bij(k)*, де *Aij(k)=A[i][j], Bij(k)=B[i][j],* *k* - індекс рекурсії.

3. Аналітичні оцінки кількості арифметичних операцій та їх порівняння

В локалізованому графі залежностей кількість операцій рівна N2(2N-1) де n кількість стовпців чи рядків матриці.(Для обчислення кожного з N2 елементів необхідно 2N-1 операцій).

4. Текст програми, що реалізовує оптимізований локально-рекурсивний алгоритм:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Pro\_lab2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

int N=0;

int[,] arrayA;

int[,] arrayB;

private void maskedTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == Convert.ToChar(13))

{

N = Int32.Parse(maskedTextBox1.Text);

if (N > 13)

{

MessageBox.Show("Матриця завелика для коректного відображення");

return;

}

arrayA = new int[N, N];

arrayB = new int[N, N];

int temp = 0, temp2 = 0;

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

{

arrayA[j, j] = (j+1)\*(j+2);

temp2 = temp2 + 2;

}

Random rnd = new Random(DateTime.Now.Millisecond);

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

{

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

arrayB[i,j] = 0;

}

int k = N;

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

{

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

{

arrayB[i,j] = rnd.Next(9) + 1;

}

k--;

}

Show(arrayA, N, textBox1,true);

Show(arrayB, N, textBox2,false);

}

}

private void Show(int[,] arr,int N, TextBox txt, bool flag)

{

string probil = " ";

string temp = "";

txt.Clear();

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

{

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

{

if (i == j && flag)

{

txt.Text += arr[i, j].ToString();

switch (arr[i, j].ToString().Length)

{

case 1:

temp = probil.Remove(0, probil.Length - 4);

txt.Text += temp;

break;

case 2:

temp = probil.Remove(0, probil.Length - 3);

txt.Text += temp;

break;

case 3:

temp = probil.Remove(0, probil.Length - 2);

txt.Text += temp;

break;

case 4:

temp = probil.Remove(0, probil.Length - 1);

txt.Text += temp;

break;

}

}

else

{

txt.Text += arr[i, j].ToString() + probil;

}

}

txt.Text += Environment.NewLine;

}

}

private void button1\_Click(object sender, EventArgs e)

{

Count = 0;

textBox3.Clear();

if (N != Int32.Parse(maskedTextBox1.Text))

{

KeyPressEventArgs e1 = new KeyPressEventArgs(Convert.ToChar(13));

maskedTextBox1\_KeyPress(sender, e1);

}

ArrayMul(arrayA, arrayB, N);

localAlgorutm(arrayA, arrayB, N);

}

int Count = 0;

void ArrayMul(int[,] arrA, int[,] arrB, int N)

{

int[,,] arrC = new int[N, N, N+1];

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

{

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

{

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

{

arrC[i, j, k + 1] = arrC[i, j, k] + arrA[i, k] \* arrB[k, j];

Count += 2;

}

Count--;

}

}

Show(arrC, N,"З одноразовим присвоєнням", Count);

}

private void localAlgorutm(int[,] arrA, int[,] arrB, int N)

{

int q = 0;

int[, ,] TempA = new int[N, N, N + 1];

int[, ,] TempB = new int[N, N, N + 1];

int[, ,] arrC = new int[N, N, N + 1];

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

{

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

{

TempA[i, j, 0] = arrA[i, j];

TempB[i, j, 0] = arrB[i, j];

}

}

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

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

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

{

TempA[i, k, j + 1] = TempA[i, k, j];

TempB[k, j, i + 1] = TempB[k, j, i];

if (TempA[i, k, 0] != 0 && TempB[k, j, 0] != 0)

{

arrC[i, j, k + 1] = arrC[i, j, k] + TempA[i, k, j] \* TempB[k, j, i];

q += 2;

}

else arrC[i, j, k + 1] = arrC[i, j, k];

}

Show(arrC, N,"Локально-рекурсивний алгоритм" ,q);

}

private void Show(int[, ,] arrC, int N, string Mess, int q)

{

textBox3.Text += Mess + Environment.NewLine;

for (int i = 0; i < arrC.GetLength(0); i++)

{

for (int j = 0; j < arrC.GetLength(1); j++)

{

textBox3.Text += arrC[i, j, N].ToString() + "\t";

}

textBox3.Text += Environment.NewLine;

}

textBox3.Text += "----------------------------------------------------" + Environment.NewLine;

textBox3.Text += "Кiлькiсть арифметичних операцій: " + q.ToString() + Environment.NewLine;

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

}

}

**Висновок:** В роботі використано паралелізм на рівні підзадач, оскільки передбачається, що кожен блок зі схеми декомпозиції є реалізований у виді функції. Це є середньоблоковий паралелізм. Обмін даними відбувається через використання спільних змінних. Присутня залежність даних між різними рівнями декомпозиції, але в межах одного рівня її немає. Є залежність за керуванням, оскільки послідовність обчислювального процесу наперед однозначно відома. Залежність за ресурсами та вводом/виводом може бути визначена лише у відношенні до певної обчислювальної системи.