­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­



Лабораторна робота №3

з курсу «**Технології розподілених систем та паралельних обчислень**»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 11

Виконав студент гр. КНз-41

Чалий Михaйло

Перевірив

Фармага І.В.

­­

Львів 2016

## Мета роботи

Вивчити методи декомпозицій задач. Набути навиків розв’язування задач з використанням функціональної декомпозиції.

## Теоретична частина

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. *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.

**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.

**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.

**4. *Планування обчислень***. На цьому кроці виконується розподіл підзадач 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.

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

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | матриця | | |
| bi=11i2 для парних і  bi=11/i для непарних і | A1(b1-2c1) | A2(B2-2C2)  Cij =1/(i2+j) |

## Реалізація

Лістінг Program.cs- https://github.com/chaliy/studies-octo-adventure/blob/master/lp/c4\_2/trs/l3/ Program.cs

﻿using System;

using System.Threading.Tasks;

using MathNet.Numerics.LinearAlgebra;

using MathNet.Numerics.LinearAlgebra.Double;

namespace L3FunctionalDecomposition

{

public class Program

{

private class Strategy

{

readonly int n;

readonly Random rnd = new Random();

public Strategy(int n)

{

this.n = n;

}

async Task<Matrix<double>> BuildMatrix(Func<int, int, double> formula)

{

return DenseMatrix.Create(n, n, formula);

}

async Task<Vector<double>> BuildVector(Func<int, double> formula)

{

return DenseVector.Create(n, formula);

}

public async Task<Matrix<double>> Go()

{

var A = await BuildMatrix((x, y) => rnd.NextDouble());

var b = await BuildVector(i => (i % 2 == 0) ? 11 \* Math.Pow(i, 2) : 11 / i);

var y1 = A.Multiply(b).ToColumnMatrix();

Console.WriteLine($"y1: ${y1}");

var A1 = await BuildMatrix((x, y) => rnd.NextDouble());

var b1 = await BuildVector(i => rnd.NextDouble());

var c1 = await BuildVector(i => rnd.NextDouble());

var y2 = A1.Multiply(b1.Subtract(c1.Multiply(2))).ToColumnMatrix();

Console.WriteLine($"y2: ${y2}");

var A2 = await BuildMatrix((x, y) => rnd.NextDouble());

var B2 = await BuildMatrix((x, y) => rnd.NextDouble());

var C2 = await BuildMatrix((x, y) => (x + y == 0) ? 0.0 : 1 / (Math.Pow(x,2) + y));

var Y3 = A2.Multiply(B2.Subtract(C2));

Console.WriteLine($"Y3: ${Y3}");

var x1 = y2.Multiply(y1.Transpose());

var x2 = Y3.Multiply(y2.Transpose().Multiply(y2).At(0, 0));

var x3 = y1.Transpose().Multiply(Y3.Power(2)).Multiply(y2).At(0, 0);

var x4 = Y3;

var x5 = Y3.Multiply(y1).Multiply(y1.Transpose()).Multiply(Y3);

return x1.Add(x2).Add(x3).Add(x4).Add(x5);

}

}

public static void Main(string[] args)

{

var strategy = new Strategy(5);

var result = strategy.Go().Result;

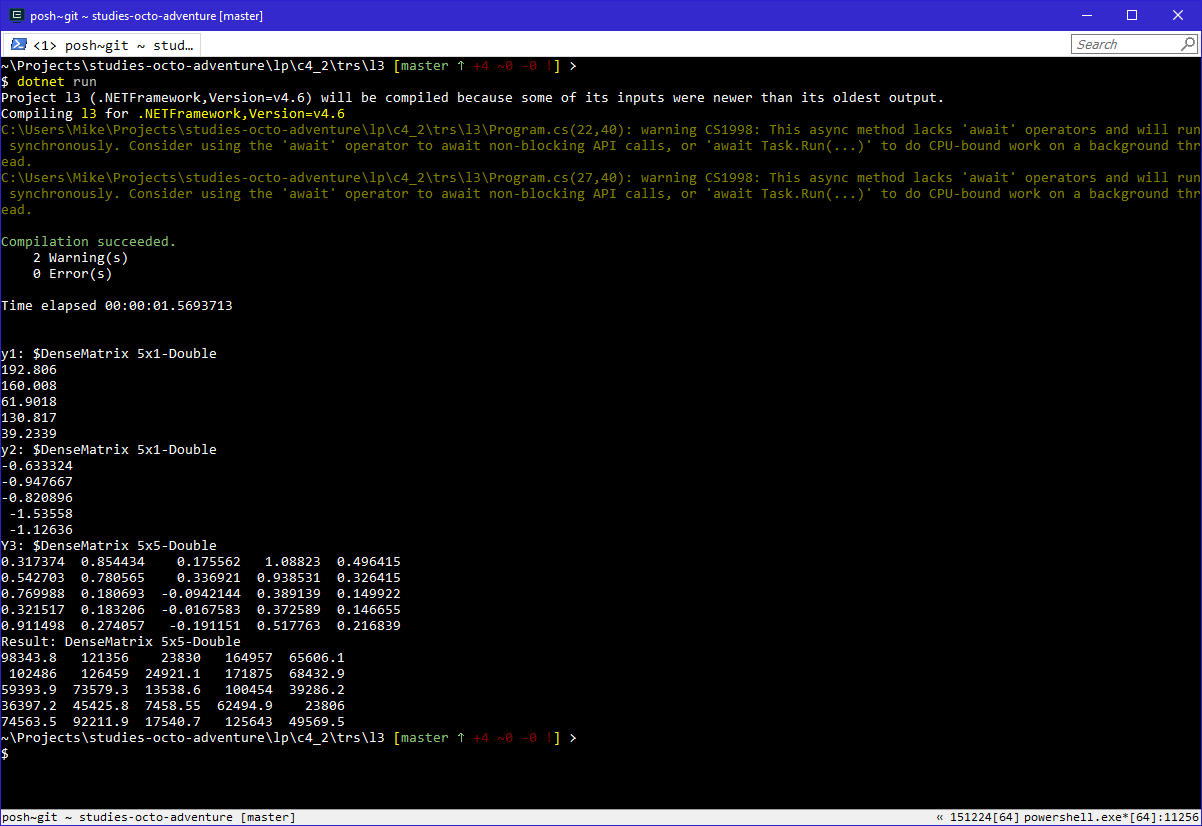
Console.WriteLine($"Result: {result}");

}

}

}

## Результат



## Висновки

Вивчив методи декомпозицій задач. Набув навиків розв’язування задач з використанням функціональної декомпозиції.