Міністерство освіти і науки України

Львівський національний університет Ім. І. Франка

Факультет прикладної математики та інформатики

Кафедра дискретного аналізу та інтелектуальних систем

Паралельні та розподілені обчислення

Лабораторна робота №2

Множення матриць

Роботу виконала:

Студентка ПМІ-33

Багінська Маргарита

Прийняв:

доц. Пасічник Т.В.

Львів 2023

Тема: Множення матриць.

Мета: Реалізувати послідовний та паралельний алгоритм множення двох матриць.

Послідовний алгоритм

```
public static TimeSpan SyncMethod(int[,] matrixA, int[,] matrixB)
{
    int rows1 = matrixA.GetLength(0);
    int cols1 = matrixA.GetLength(1);
    int rows2 = matrixB.GetLength(0);
    int cols2 = matrixB.GetLength(1);
    if (cols1 != rows2)
    {
        throw new ArgumentException("Matrices cannot be multiplied.");
    }
    int[,] result = new int[rows1, cols2];

    Stopwatch stopWatch = new Stopwatch();
    stopWatch.Start();
    for (int i = 0; i < rows1; i++)
    {
        for (int j = 0; j < cols2; j++)
        {
            int sum = 0;
            for (int k = 0; k < cols1; k++)
            {
                sum += matrixA[i, k] * matrixB[k, j];
            }
            result[i, j] = sum;
        }
    }
    stopWatch.Stop();
    Console.WriteLine("Sync time ~ " + stopWatch.Elapsed.ToString());
    return stopWatch.Elapsed;
      }
```

У послідовному алгоритмі проходимось по рядку першої матриці та по стовпцю другої матриці двома вкладеними циклами, третім циклом рахуємо результат множення рядка на стовпець та у результуючу матрицю, засікаємо час.

Паралельний алгоритм

```
public static TimeSpan AsyncMethod(int[,] matrixA, int[,] matrixB, int threadNum)
{
    int rowsA = matrixA.GetLength(0);
    int colsA = matrixA.GetLength(1);
    int rowsB = matrixB.GetLength(0);
    int colsB = matrixB.GetLength(1);

    if (colsA != rowsB)
    {
        throw new ArgumentException("Matrices cannot be multiplied.");
    }

    int[,] result = new int[rowsA, colsB];

    int totalElements = rowsA * colsB;
    int elementsPerThread = totalElements / threadNum;

    List<Thread> threads = new List<Thread>();

    Stopwatch stopWatch = new Stopwatch();
    stopWatch.Start();
    int start = 0;
    for (int i = 0; i < threadNum; i++)
    {
        int end = (i == threadNum - 1) ? totalElements : start + elementsPerThread;

        Thread thread = new Thread((object range) =>
        {
            int[] rangeArray = (int[])range;
            int startRange = rangeArray[0];
            int endRange = rangeArray[1];

            for (int index = startRange; index < endRange; index++)
            {
                int row = index / colsB;
                int col = index % colsB;

                int sum = 0;
                for (int k = 0; k < colsA; k++)
                {
                    sum += matrixA[row, k] * matrixB[k, col];
                }
                result[row, col] = sum;
            }
        });

        threads.Add(thread);
        thread.Start(new int[] { start, end });

        start = end;
    }

    foreach (var thread in threads)
    {
        thread.Join();
    }
    stopWatch.Stop();

    Console.WriteLine($"Async time ~ {stopWatch.Elapsed} with {threadNum} threads");
    return stopWatch.Elapsed;
      }
```

У розпаралеленому алгоритмі множення двох матриць за допомогою потоків Threads ми використовуємо цикл for, щоб розділити обчислення елементів матриці між потоками. Для кожного

потоку він обчислює діапазон елементів для обробки на основі початкової та кінцевої змінних. Усередині потоку проходимо по діапазону елементів і обчислюємо відповідний елемент.

Виконуємо множення для кожного елемента та зберігаємо результат у відповідному рядку та стовпці результату.

Переконаємось у коректності роботи:

```
Matrix A:
9 1 6
3 8 2

Matrix B:
4 4
8 6
3 6

Result:
62 78
82 72
```

Прискорення Sp для паралельного алгоритму визначається відношенням часової складності послідовного T1 та паралельного алгоритмів для p процесорів Sp = T1 / Tp. (Sp > 1 Оптимально).

Ефективність Ep для паралельного алгоритму визначається прискоренням цього алгоритму відносно кількості процесорів: Ep= Sp/p Ідеал: Ep(n) = 1.

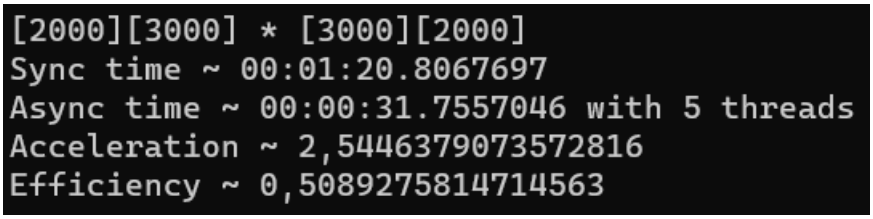
Результати

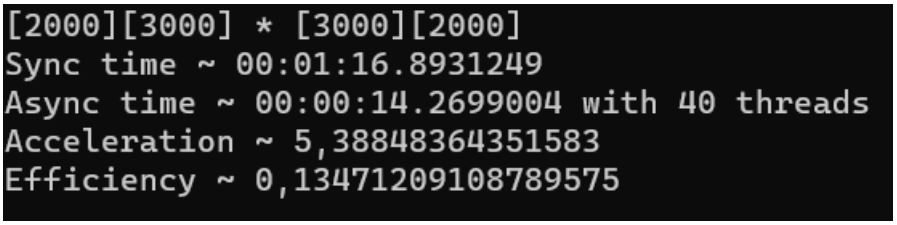
```
[20][30] * [30][20]
Sync time ~ 00:00:00.0003258
Async time ~ 00:00:00.0312782 with 4 threads
Acceleration ~ 0,010416200420740323
Efficiency ~ 0,002604050105185081
```

На малих розмірностях матриць розпаралелення не є оптимальним.

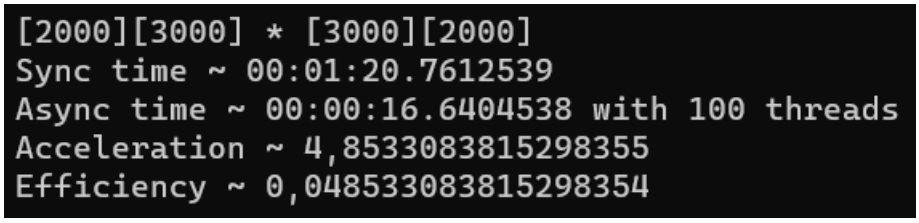
```
[2000][3000] * [3000][2000]
Sync time ~ 00:01:17.8692837
Async time ~ 00:00:51.6082704 with 3 threads
Acceleration ~ 1,508852807824383
Efficiency ~ 0,5029509359414 61
```

Зі збільшенням розмірності паралельність ефективніша.





Можемо спостерігати ефективність паралельного алгоритму при великих об’ємах та розумній кількості потоків.



Висновок: У результаті виконання лабораторної роботи було реалізовано послідовний та паралельний алгоритм множення двох матриць мовою програмування C# та класу Thread. Переконались у високій ефективності розпаралелення процесу у даному випадку.