**Міністерство освіти і науки України**
**Львівський національний університет імені Івана Франка**
Факультет прикладної математики та інформатики

Звіт

Лабораторна робота №2

Тема: «Розпаралелення множення матриць»

**з дисципліни "Паралельні та розподільні обчислення"**

Виконав студент групи ПМі-31
Яцуляк Андрій

Львів 2023 р.

**Мета:** Написати програми обчислення суми та різниці двох матриць (послідовний та паралельний алгоритми).

## Теоретичний матеріал

**Матриця** - це структура даних у математиці та програмуванні, яка представляє собою двовимірний масив чисел або елементів. Матриця складається з рядків та стовпців, і кожен елемент матриці розташований у певному рядку та стовпці.

Основні операції з матрицями: додавання матриць однакових розмірів, множення матриці на число, множення матриць, транспонування матриці.

## Хід роботи

Завдання виконав мовою програмування Java у середовищі IntelliJ IDEA. Написав програму для роботи з матрицями:

```java
public class Matrix
{
    25 usages
    private int rows;
    27 usages
    private int columns;
    11 usages
    private final int[][] matrix;
    2 usages
    private static int theardsNumber;

    8 usages
    Matrix(int rows, int columns)
    {...}
    no usages
    Matrix(int[][] matrix) { this.matrix = matrix; }
    2 usages
    public void fillMatrix()
    {...}
    4 usages
    public void printMatrix()
    {...}
    no usages
    public int getRows() { return this.rows; }
    no usages
    public void setRows(int rows) { this.rows = rows; }
    no usages
    public int getColumns() { return this.columns; }
```

```
52      public void setColumns(int columns) { this.columns = columns; }
        3 usages
56      public static int getTheardsNumber() { return Matrix.theardsNumber; }
        1 usage
60      public static void setTheardsNumber(int theardsNumber) { Matrix.theardsNumber = theardsNumber; }
        7 usages
64      public int getElement(int rows, int columns) { return this.matrix[rows][columns]; }
        6 usages
68      public void setElement(int rows, int columns, int value) { this.matrix[rows][columns] = value; }
        no usages
72  @   public Matrix addMatrix(Matrix other)
73      {...}
        no usages
88  @   public Matrix addMatrixByParallel(Matrix other) {...}
        no usages
115 @   public Matrix subtractionMatrix(Matrix other)
116     {...}
        no usages
131 @   public Matrix subtractionMatrixByParallel(Matrix other)
132     {...}
        1 usage
159 @   public Matrix multiplyMatrix(Matrix other)
160     {...}
        1 usage
181 @   public Matrix multiplyMatrixByParallel(Matrix other) {...}
213 }
```

## Множення матриць

Для початку, я створив і заповнив дві матриці розміром 3x3 для того, щоб переконатись у коректності послідовного та паралельного множення:

```
matrix1.fillMatrix();
matrix1.setTheardsNumber(numThreads);
matrix1.printMatrix();
System.out.print("------------------------------\n");
matrix2.fillMatrix();
matrix2.printMatrix();
System.out.print("---------------Result---------------\n");
Matrix resultMatrix = matrix1.multiplyMatrix(matrix2);
resultMatrix.printMatrix();
System.out.print("---------------Result2---------------\n");
Matrix resultMatrix2 = matrix1.multiplyMatrixByParallel(matrix2);
resultMatrix2.printMatrix();
```

```
Enter the number of rows of the matrix: 3
Enter the number of columns of the matrix: 3
Enter the number of threads: 2
8 2 0
1 8 1
4 7 10
------------------------------
10 2 9
6 8 6
9 6 0
---------------Result---------------
92 32 84
67 72 57
172 124 78
---------------Result2---------------
92 32 84
67 72 57
172 124 78
```

Далі створив дві матриці розміром 1000x1000, заповнив випадковими числами від 0 до 1000 методом fillMatrix. Також створив ще дві додаткові матриці, для зберігання результату послідовного та паралельного множення. Провів обрахунки на різній кількості заданих потоків. Обчислив час виконання послідовного та паралельного множення, а також обчислив прискорення та ефективність. Далі наведені скріни консолі:

```
Enter the number of rows of the matrix: 1000
Enter the number of columns of the matrix: 1000
Enter the number of threads: 2

Time taken sequential: 1419930101 nanoseconds
Time taken parallel: 666105299 nanoseconds
Speedup: 2.1316901443836134
Efficiency: 1.0658450721918067
```

```
Enter the number of rows of the matrix: 1000
Enter the number of columns of the matrix: 1000
Enter the number of threads: 3

Time taken sequential: 1358073400 nanoseconds
Time taken parallel: 473188300 nanoseconds
Speedup: 2.8700485620629252
Efficiency: 0.9566828540209751
```

```
Enter the number of rows of the matrix: 1000
Enter the number of columns of the matrix: 1000
Enter the number of threads: 4

Time taken sequential: 1564345000 nanoseconds
Time taken parallel: 421426901 nanoseconds
Speedup: 3.7120197981855934
Efficiency: 0.9280049495463983
```

```
Enter the number of rows of the matrix: 1000
Enter the number of columns of the matrix: 1000
Enter the number of threads: 10

Time taken sequential: 1576300400 nanoseconds
Time taken parallel: 325269400 nanoseconds
Speedup: 4.846138001299845
Efficiency: 0.48461380012998456
```

```
Enter the number of rows of the matrix: 1000
Enter the number of columns of the matrix: 1000
Enter the number of threads: 11

Time taken sequential: 1587381300 nanoseconds
Time taken parallel: 286766401 nanoseconds
Speedup: 5.535450786649165
Efficiency: 0.5032227987862877
```

**Висновок.** Під час виконання лабораторної роботи я написав програму для роботи з матрицями, зокрема додавання матриць послідовним та паралельними алгоритмами, обчислив прискорення та ефективність для різної кількості потоків.