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

Звіт

Лабораторна робота №1

Тема: «Розпаралелення додавання/віднімання 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 }
```

Додавання матриць

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

```
17          Matrix matrix1 = new Matrix(rows, columns);
18          Matrix matrix2 = new Matrix(rows, columns);
19
20          matrix1.fillMatrix();
21          matrix1.setTheardsNumber(numThreads);
22          matrix1.printMatrix();
23          System.out.print("------------------------------\n");
24          matrix2.fillMatrix();
25          matrix2.printMatrix();
26          System.out.print("---------------Result1---------------\n");
27          Matrix resultMatrix = matrix1.addMatrix(matrix2);
28          resultMatrix.printMatrix();
29          System.out.print("---------------Result2---------------\n");
30          Matrix resultMatrix2 = matrix1.addMatrixByParallel(matrix2);
31          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
827 260 427
801 494 669
443 753 31
------------------------------
742 784 735
253 947 29
533 500 371
---------------Result1---------------
1569 1044 1162
1054 1441 698
976 1253 402
---------------Result2---------------
1569 1044 1162
1054 1441 698
976 1253 402
```

Далі створив дві матриці розміром 10000x10000, заповнив випадковими числами від 0 до 1000 методом fillMatrix. Також створив ще дві додаткові матриці, для зберігання результату послідовного та паралельного додавання. Провів обрахунки на різній кількості заданих потоків. Обчислив час виконання послідовного та паралельного додавання, а також обчислив прискорення та ефективність. Далі наведені скріни консолі:

```
Enter the number of rows of the matrix: 10000
Enter the number of columns of the matrix: 10000
Enter the number of threads: 2

Time taken sequential: 4809544600 nanoseconds
Time taken parallel: 3691299800 nanoseconds
Speedup: 1.3029406606312497
Efficiency: 0.6514703303156248
```

```
Enter the number of rows of the matrix: 10000
Enter the number of columns of the matrix: 10000
Enter the number of threads: 3

Time taken sequential: 4716031300 nanoseconds
Time taken parallel: 417900300 nanoseconds
Speedup: 11.285063207659817
Efficiency: 3.7616877358866057
```

```
Enter the number of rows of the matrix: 10000
Enter the number of columns of the matrix: 10000
Enter the number of threads: 4

Time taken sequential: 800005200 nanoseconds
Time taken parallel: 299746200 nanoseconds
Speedup: 2.6689419248684385
Efficiency: 0.6672354812171096
```

```
Enter the number of rows of the matrix: 10000
Enter the number of columns of the matrix: 10000
Enter the number of threads: 10

Time taken sequential: 898941400 nanoseconds
Time taken parallel: 312333000 nanoseconds
Speedup: 2.8781505636612206
Efficiency: 0.28781505636612204
```

```
Enter the number of rows of the matrix: 10000
Enter the number of columns of the matrix: 10000
Enter the number of threads: 11

Time taken sequential: 340640200 nanoseconds
Time taken parallel: 316111800 nanoseconds
Speedup: 1.077594066403089
Efficiency: 0.09796309694573536
```

Найкращий результат вийшов при кількості потоків 3, найгірший при 11.

Також обчислив послідовне та паралельне віднімання двох матриць 10000x10000:

```
Enter the number of rows of the matrix: 10000
Enter the number of columns of the matrix: 10000
Enter the number of threads: 2

Time taken sequential: 271852000 nanoseconds
Time taken parallel: 309705600 nanoseconds
Speedup: 0.8777755390926092
Efficiency: 0.4388877695463046
```

```
Enter the number of rows of the matrix: 10000
Enter the number of columns of the matrix: 10000
Enter the number of threads: 3

Time taken sequential: 473547800 nanoseconds
Time taken parallel: 300823000 nanoseconds
Speedup: 1.5741741821602737
Efficiency: 0.5247247273867579
```

```
Enter the number of rows of the matrix: 10000
Enter the number of columns of the matrix: 10000
Enter the number of threads: 4

Time taken sequential: 867096000 nanoseconds
Time taken parallel: 323631200 nanoseconds
Speedup: 2.6792719614178115
Efficiency: 0.6698179903544529
```

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.