Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Звіт до лабораторної роботи №1

«Програмування потоків. Потоки в мові Java»

з дисципліни **«Програмне забезпечення високопродуктивних комп'ютерних систем»**

Виконала:
студентка групи ІМ-13
Мартинюк Марія Павлівна
номер у списку групи 13

Перевірив:
Корочкін О. В.

Київ 2024

Мета роботи: вивчення засобів мови Java для роботи с потоками.

Виконання роботи: Розробити програму, яка містить паралельні потоки, що реалізують відповідну функцію F1, F2, F3 з Додатку Б згідно отриманому варіанту. Програма повинна мати пакет Data і основну процедуру Lab1. Пакет містить ресурси, необхідні для обчислення функцій F1, F2, F3 через підпрограми Func1, Func2, Func3.

Дослідити при виконанні програми:
- вплив пріоритетів задач на чергу запуску задач
- завантаження центрального багатоядерного процесора паралельної комп'ютерній системи (ПКС). Зміна кількості ядер здійснюється за допомогою Менеджера (Диспетчера) задач ОС Windows.

В потоках використати метод **join()**.

Варіант:
F1: E = A + C * (MA * ME) + B
F2: MF = MG * (MK * ML) - MK
F3: O = SORT(P) * (MR * MS)

Опис програми:

Для виконання роботи була обрана мова програмування Java. Причинами вибору саме цього варіанту є структурованість мови та зручність засобів для реалізації багатопоточності. Через особливість мови, що являє собою використання окремих класів та методів для написання функціоналу, програмні компоненти можуть бути реалізовані цілком незалежно один від одного. Таким чином логіку виконання обчислень потоком можна відділити від логіки самого потоку. Це сприяє оптимізації коду, його зрозумілості, зручності та швидкості у перевикористані.

Програма складається з чотирьох класів: головного класу Lab1, класів-потоків T1_F1, T2_F2, T3_F3 та класу ресурсів Data, у якому знаходяться допоміжні методи для введення даних, запису та читання файлів, а також обчислення визначених за варіантом функцій.

Запустивши програму, користувачу пропонується ввести ціле значення N, відповідно до якого в подальшому буде обраний один з 3 сценаріїв виконання обчислень. У випадку N = 4 відбувається ввід значень елементів векторів та матриць з клавіатури. Якщо ж N > 1000, зчитування даних відбувається з використанням файлової системи. Користувачу пропонується обрати або створення файлів із випадковими значеннями для змінних відповідної функції, натиснувши 1, або створення файлів із значеннями рівними 1, 2 або 3 в залежності від номера потоку та відповідної йому функції обрахунку, натиснувши 2. В будь-якому випадку далі від користувача вимагається ввести назви файлів для створення та зчитування, після чого виконуються відповідні арифметичні дії. Остаточний результат обчислення записується у файл за допомогою виклики методу `writeResultToFile()` з класу ресурсів та виводиться у консоль.

У кожному файлі потоку існує два методи: перший `inputAndCalculate()` для введення значень елементів та перевизначений метод `run()` для визначення поведінки потоку при його запуску. Варто відмітити, що програма сигналізує користувачеві про стан потоків (запущений/завершений) відповідними повідомленнями у консоль. Обробка виключних ситуацій виконується за допомогою використання структури `try-catch` з виводом повідомлень про помилки, у разі їх виникнення.

Запуск потоків відбувається в головному файлі Lab1 за допомогою створення екземплярів класів потоків та переведенням їх до стану виконання методом `start()`, що в свою чергу відповідно викликає перевизначений раніше метод `run()`. Також на кожному з екземплярів потоків викликається метод `join()`, що синхронізує головний потік програми з новоствореними екземплярами інших потоків. Таким чином забезпечується виконання головного потоку після завершення виконання інших потоків.

Пізніше додатково було додано змінні `start, end, totalTime` для відслідковування часу виконання програми та подальшого обрахування коефіцієнту прискорення при різній кількості використаних ядер процесора.

Проблеми, що виникли при виконанні дослідження:

При спробі запустити кожний потік окремо, обчислення та читання/запис файлів з даними виконуються коректно. Нижче на скріншотах показаний вивід відповідних результатів у консоль.

```
T1 is started
Enter N: 4
Enter A values (separated by spaces): 1 1 1 1
Enter B values (separated by spaces): 1 1 1 1
Enter C values (separated by spaces): 1 1 1 1
Enter MA values:
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
Enter ME values:
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
E = [18, 18, 18, 18]
T1 is finished

Process finished with exit code 0
```

вигляд результату виконання T1 при N = 4

```
T2 is started
Enter N: 4
Enter MK values:
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
Enter ML values:
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
Enter MG values:
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
MF = [[126, 126, 126, 126], [126, 126, 126, 126], [126, 126, 126, 126], [126, 126, 126, 126]]
T2 is finished

Process finished with exit code 0
```

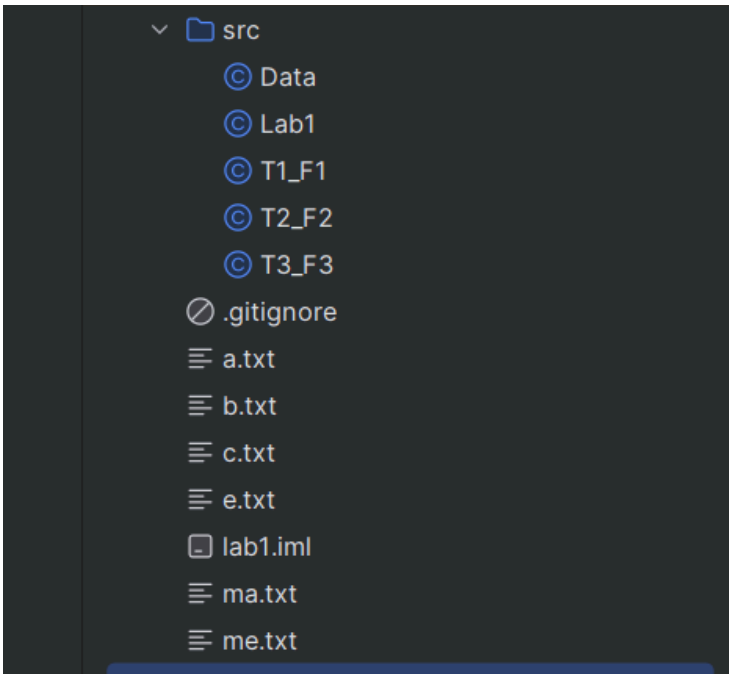
вигляд результату виконання T2 при N = 4

```
T3 is started
Enter N: 4
Enter P values (seperated by spaces): 3 3 3 3
Enter MR values:
3 3 3 3
3 3 3 3
3 3 3 3
3 3 3 3
Enter MS values:
3 3 3 3
3 3 3 3
3 3 3 3
3 3 3 3
O = [432, 432, 432, 432]
T3 is finished

Process finished with exit code 0
```

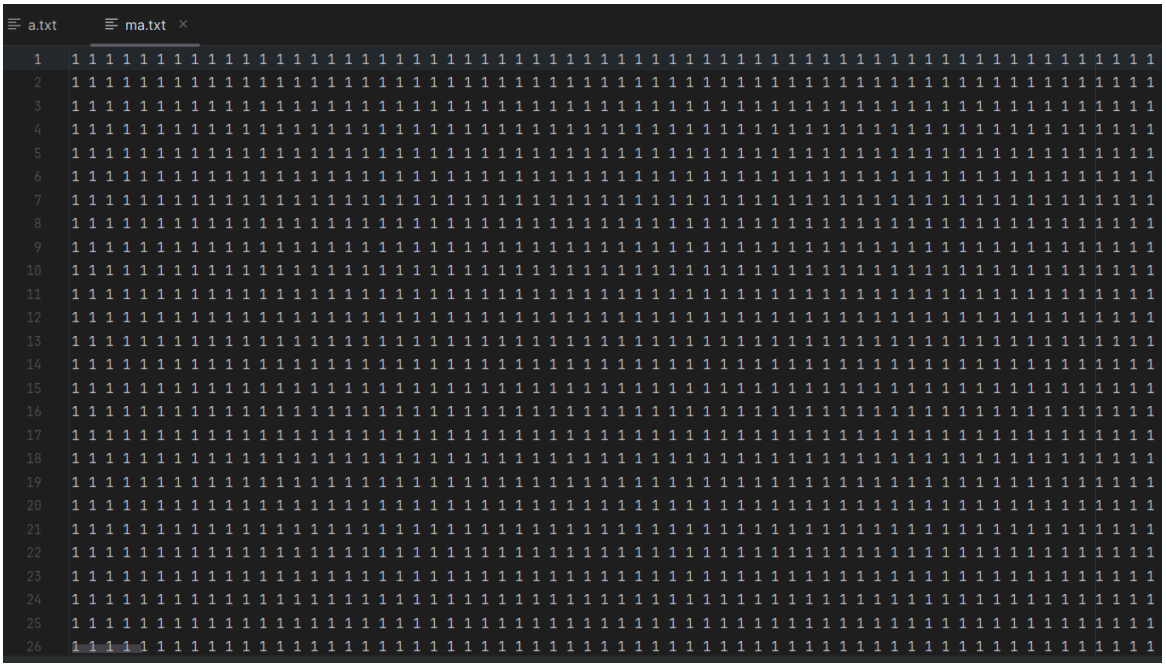
вигляд результату виконання T3 при N = 4

```
T1 is started
Enter N: 1001
Vector and matrix values are to be read from file.
Press 1 to generate random values or 2 for them to be equal 1:
2
Enter file name for A vector:
a.txt
Enter file name for B vector:
b.txt
Enter file name for C vector:
c.txt
Enter file name for MA matrix:
ma.txt
Enter file name for ME matrix:
me.txt
Enter file name for E (final result) vector:
e.txt
E = [1002003, 1002003, 1002003, 1002003, 1002003, 1002003, 1002003, 1002003, 1002003, 1002003, 1002003, 1002003,
T1 is finished

Process finished with exit code 0
```

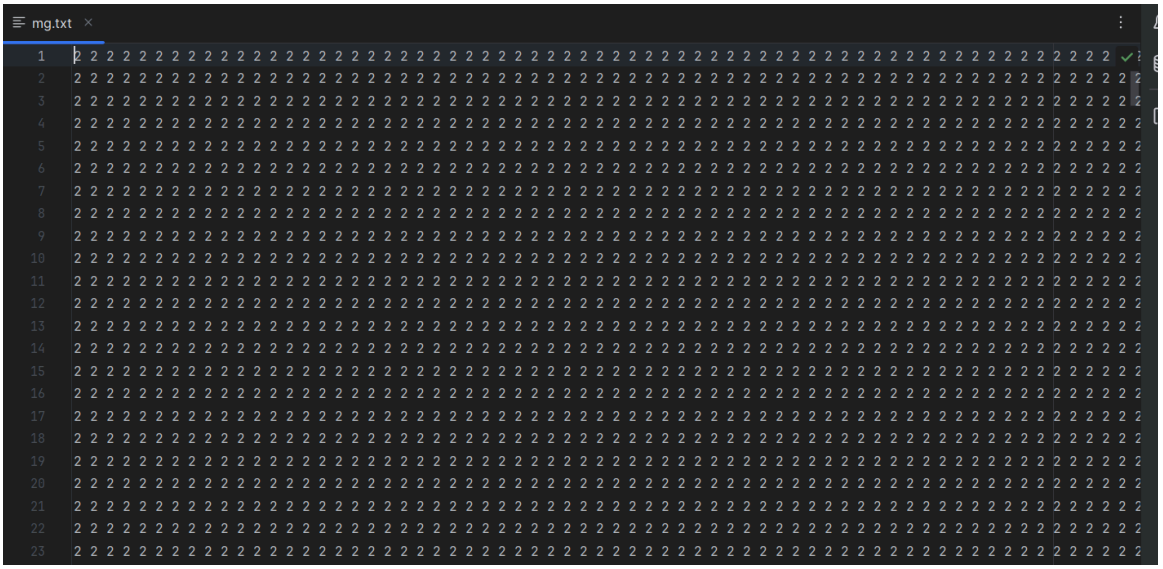
вигляд результату виконання T1 при N = 1001



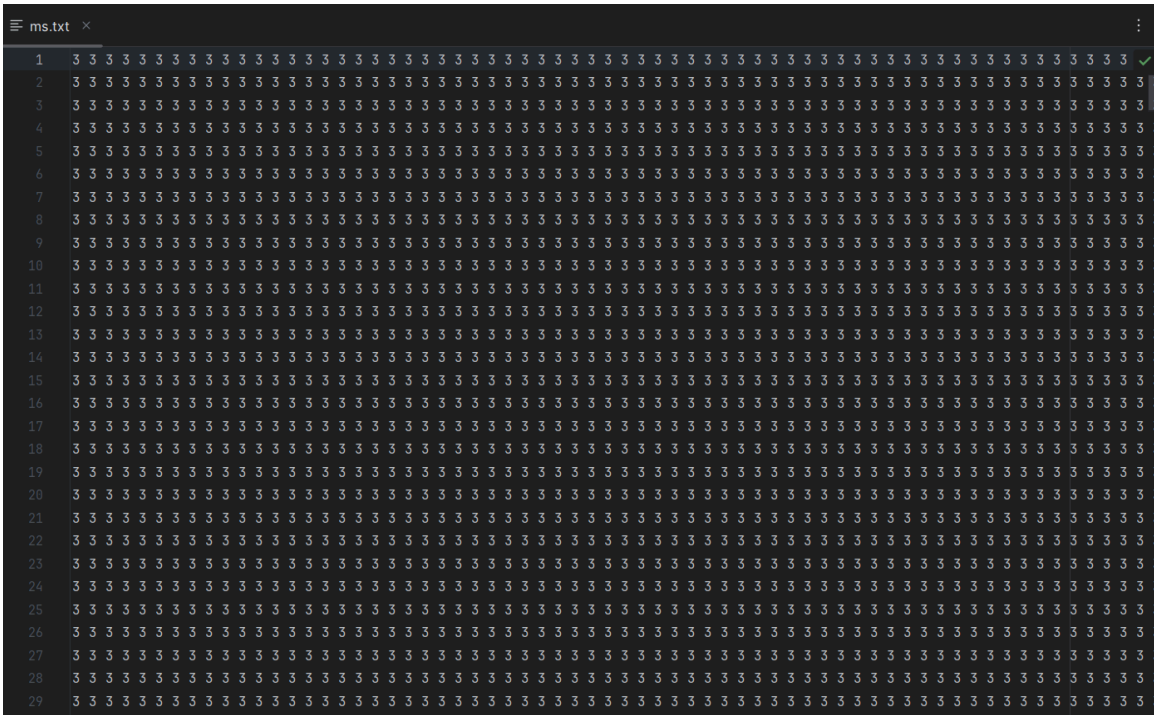
відповідно створені у директорії файли



вміст створеного T1 файлу матриці при N = 1001



вміст створеного T2 файлу матриці при N = 1001



вміст створеного Т3 файлу матриці при N = 1001

Створення та обробка файлів у варіанті режиму заповнення випадковими значеннями відбувається аналогічно коректно.

Втім, при запуску усіх трьох потоків одночасно, виникають проблеми із запитом даних. Взаємодіючи із одним джерелом виводу (консоллю) та будучи при цьому, як і вимагає технічне завдання, повністю незалежними, три паралельні програми одночасно запитують параметри для своїх подальших обчислень. Так з скріншотів нижче видно, що одразу після старту запит на параметр N викликається тричі.

```
T1 is started
T3 is started
T2 is started
Enter N: Enter N: Enter N: 4
4
Enter P values (seperated by spaces): 3 3 3 3
Enter A values (separated by spaces): 1 1 1 1
ERROR: null
ERROR: null
O = null
T3 is finished
MF = null
T2 is finished
E = null
T1 is finished

Process finished with exit code 0
```

приклад запуску 3 потоків одночасно №1

```
T1 is started
T2 is started
T3 is started
Enter N: Enter N: Enter N: 444
4 4
Enter A values (separated by spaces): 1 1 1 1
MF = null
T2 is finished

O = null
T3 is finished
ERROR: null
E = null
T1 is finished

Process finished with exit code 0
```

приклад запуску 3 потоків одночасно №2

приклад запуску 3 потоків одночасно №3

Надалі стає незрозуміло яке значення відноситься до якого запиту і які дані коли потрібно вводити. Через такі паралельні запити пізніше виникають колізії у обрахунках і потоки завершують своє виконання з невизначеним (null) фінальним результатом.

Причиною проблеми введення даних та як наслідок неправильних розрахунків є те, що запити відбуваються паралельно, а ввести відповідне значення ми можемо лише послідовно. Часто не вдається навіть ввести усі три значення N для усіх трьох потоків, одразу відбувається запит на дані для наступних кроків обрахунків, потоки яких вже отримали першочергове значення N. Можна порівняти цю ситуацію з перегонами: передбачити який з незалежних невпорядкованих між собою потоків першим надішле запит чи повідомлення у консоль практично неможливо. Також при отриманні свого значення потік може не одразу перейти до свого наступного кроку, іншими словами підвиснути. У такому випадку операційна система вивільняє потужність процесора, що займався таким потоком, та перепризначає на нього задачу іншого потоку, яка вже очікує на виконання. Це ще більше ускладнює коректність введення значень у такому варіанті програми. Ймовірним розв’язком такої проблеми може бути використання спеціальних засобів, таких як семафори, м’ютекси, захоплювачів або інших модифікаторів для контролю доступу до змінних та синхронізації багатопоточності.

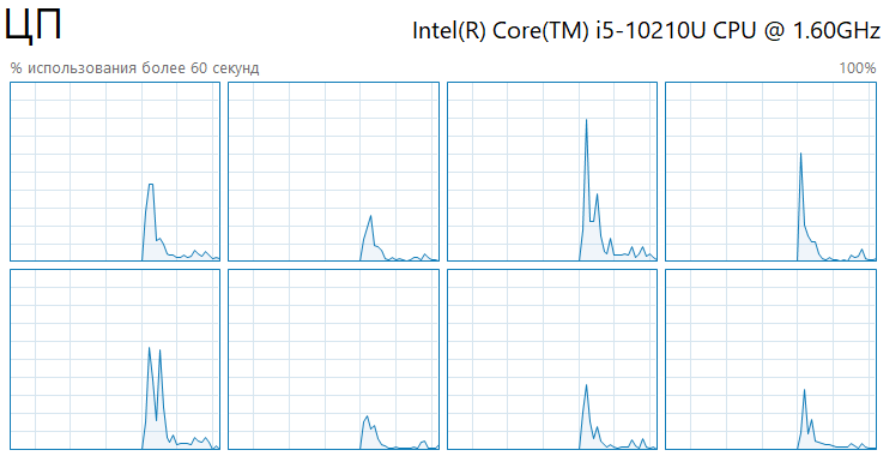
Також варто відмітити особливість запуску потоків. Через відсутність вказаного параметра пріоритету і використання його значення за замовчуванням, при кожному новому запуску основної програми черга потоків різна. До прикладу, на другому скріншоті вище спочатку почав роботу перший потік, потім третій, а лише після них другий. Черга

завершення потоків залежить від складності виконуваних математичних дій. Так за моїм варіантом в теорії останнім завершив би роботу перший потік, бо він має найбільшу кількість операцій у власній формулі розрахунку з-поміж усіх трьох. На практиці так і є.

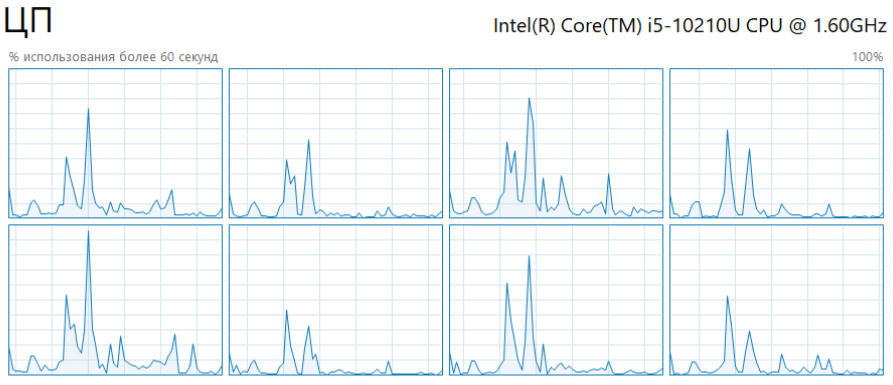
Розглядаючи варіант при N > 1000, проблеми ті ж самі.

Тестування завантаженості ядер процесора:

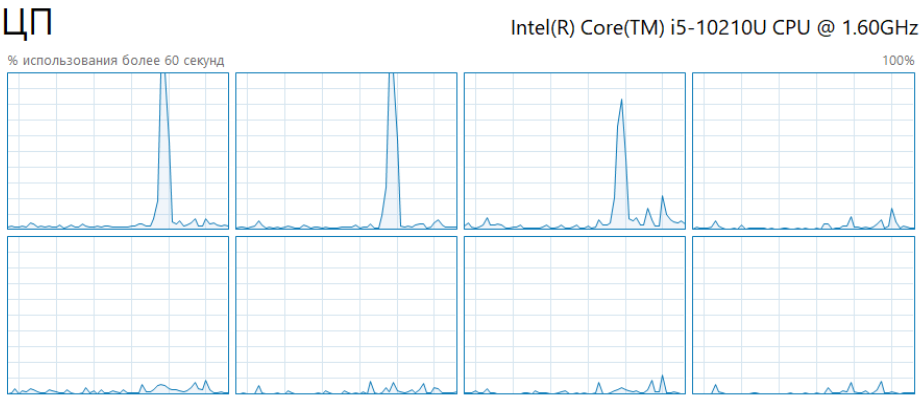
До запуску програми діаграми завантаженості виглядала так. Варто зазначити, що конкретно мій комп'ютер містить 8 логічних ядер та 4 фізичних. Усі логічні ядра першочергово використовувались більш-менш рівномірно.



Наступний скріншот ілюструє завантаженість цієї ж кількості ядер при виконанні паралельної багатопотокової програми при N = 1001.



Далі у Диспетчері задач ОС було зменшено кількість використаних середовищем розробки ядер процесора до 3. На скріншоті нижче проілюстровано, що показники завантаженості значно зросли. Пік графіку відповідає за початок виконання програми, допоки не виникли колізії через паралельне невпорядковане виконання незалежних потоків.



Було проведено ще декілька дослідів, результати яких показали, що при використанні 8 ядер пікова завантаженість була рівна 36%. При використанні 3 ядер пікова завантаженість була рівна 47%.

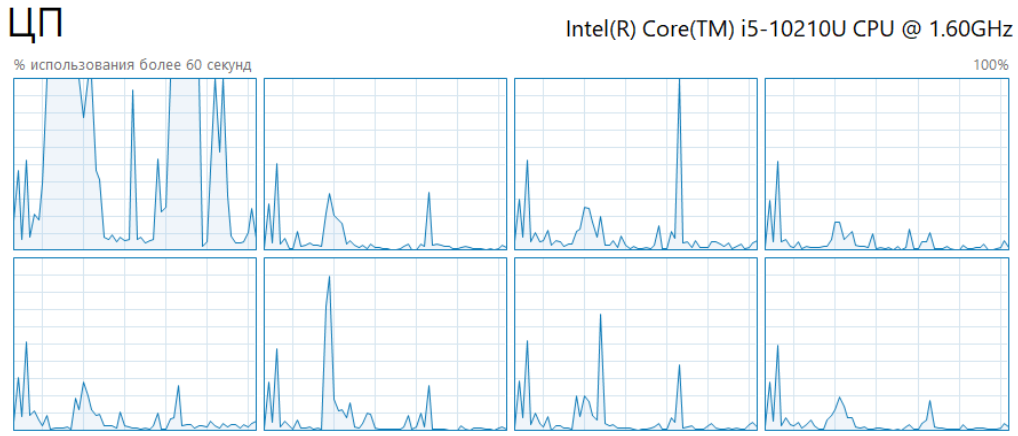
Окрім цього було додатково досліджено повний час виконання програми T2_F2 на 4 ядрах та 1 ядрі. Час виконання у першому випадку відповідно дорівнює наступному значенню:

```
 8016006, 8016006, 8016006, 8016006, 8016006, 8016006,
 8016006, 8016006, 8016006, 8016006, 8016006]]
T2 is finished
TIME 20969.0ms

Process finished with exit code 0
```

Час виконання у другому випадку відповідно дорівнює:

```
 8016006, 8016006, 8016006, 8016006, 80
 8016006, 8016006, 8016006, 8016006, 80
T2 is finished
TIME 32344.0ms

Process finished with exit code 0
```

Графік завантаженості ядер при виконанні цього досліду має наступний вигляд:



Коефіцієнт прискорення: Ку = 32344 / 20969 = 1.542. Тобто при використанні 4 ядер замість 1 програма завершується у 1.5 рази швидше.

Таким чином можна впевнитись, що при використанні меншої кількості ядер процесора, навантаженння на кожне з них зростає і швидкість виконання програми спадає.

Лістинг програми:

Lab1.java

```
//Дисципліна "Програмне забезпечення високопродуктивних комп'ютерних систем"
//Лабораторна робота ЛР1
//F1: E = A + C * (MA * ME) + B
//F2: MF = MG * (MK * ML) - MK
//F3: O = SORT(P) * (MR * MS)
//Мартинюк Марія Павлівна
//Дата 17.02.2024

public class Lab1 {
   public static void main(String[] args) throws
InterruptedException {
        //variables to estimate process time
        double start, end, totalTime;

        start = System.currentTimeMillis();
```

```
        T1_F1 T1 = new T1_F1();
        T2_F2 T2 = new T2_F2();
        T3_F3 T3 = new T3_F3();

        //threads start
        T1.start();
        T2.start();
        T3.start();

        try {
            T1.join();
            T2.join();
            T3.join();
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }

        end = System.currentTimeMillis();;
        totalTime = end - start;

        System.out.println("TIME " + totalTime + "ms");

    }
}
```

Data.java

```
import java.util.Arrays;
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Random;

public class Data {
    //formula 1 for T1
    public static int[] F1(int[] A, int[] B, int[] C, int[][] MA,
int[][] ME) {
        //E = A + C * (MA * ME) + B

        //Step 1: MA * ME
        int[][] resultMatrixMultiply = multiplyMatrices(MA, ME);

        //Step 2: C * (MA * ME)
```

```
        int[] intermediateResult =
multiplyMatrixWithVector(resultMatrixMultiply, C);

        //Step 3: E = A + C * (MA * ME) + B
        int[] E = new int[A.length];
        for (int i = 0; i < A.length; i++) {
            E[i] = A[i] + intermediateResult[i] + B[i];
        }
        return E;
    }

    //formula 2 for T2
    public static int[][] F2(int[][] MK, int[][] ML, int[][] MG) {
        //MF = MG * (MK * ML) - MK

        //Step 1: MK * ML
        int[][] resultMatrixMultiply = multiplyMatrices(MK, ML);

        //Step 2: MG * (MK * ML)
        int[][] resultMatrixMultiply2 = multiplyMatrices(MG,
resultMatrixMultiply);

        //Step 3: MF = MG * (MK * ML) - MK
        int[][] MF = new int[MG.length][MG[0].length];
        for (int i = 0; i < MG.length; i++) {
            for (int j = 0; j < MG[0].length; j++) {
                MF[i][j] = resultMatrixMultiply2[i][j] - MK[i][j];
            }
        }
        return MF;
    }

    //formula 3 for T3
    public static int[] F3(int[] P, int[][] MR, int[][] MS) {
        //O = SORT(P) * (MR * MS)

        //Step 1: MR * MS
        int[][] resultMatrixMultiply = multiplyMatrices(MR, MS);

        //Step 2: SORT(P)
        int[] SortP = Arrays.copyOf(P, P.length);
        Arrays.sort(SortP);

        //Step 3: O = SORT(P) * (MR * MS)
        int[] O = multiplyMatrixWithVector(resultMatrixMultiply,
SortP);
        return O;
    }
```

```
    //helping method to multiply matrices
    private static int[][] multiplyMatrices(int[][] matrixOne,
int[][] matrixTwo) {
        int rowsMatrixOne = matrixOne.length;
        int colsMatrixOne = matrixOne[0].length;
        int colsMatrixTwo = matrixTwo[0].length;

        int[][] result = new int[rowsMatrixOne][colsMatrixTwo];

        for (int i = 0; i < rowsMatrixOne; i++) {
            for (int j = 0; j < colsMatrixTwo; j++) {
                for (int k = 0; k < colsMatrixOne; k++) {
                    result[i][j] += matrixOne[i][k] *
matrixTwo[k][j];
                }
            }
        }
        return result;
    }

    //helping method to multiply matrix with vector
    private static int[] multiplyMatrixWithVector(int[][] matrix,
int[] vector) {
        int rows = matrix.length;
        int cols = matrix[0].length;

        int[] result = new int[rows];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i] += matrix[i][j] * vector[j];
            }
        }
        return result;
    }

    //helping method to input vector values via keyboard
    public static int[] vectorInput(int N, Scanner scan) {
        int[] vector = new int[N];
        for (int i = 0; i < N; i++) {
            vector[i] = scan.nextInt();
        }
        return vector;
    }

    //helping method to input matrix values via keyboard
    public static int[][] matrixInput(int N, Scanner scan) {
        int[][] matrix = new int[N][N];
```

```
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                matrix[i][j] = scan.nextInt();
            }
        }
        return matrix;
    }

    //helping method to write and read a file for matrix elements
    public static int[][] matrixCreate(int N, boolean
generateRandom, Scanner scan, int inputNumberParam) {
        String filename = scan.nextLine();
        int[][] matrix = new int[N][N];

        //write
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filename))) {
            for (int i = 0; i < matrix.length; i++) {
                for (int j = 0; j < matrix[i].length; j++) {
                    int value = generateRandom ? new
Random().nextInt(10) : inputNumberParam;
                    writer.write(value + " ");
                }
                writer.newLine();
            }
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage());
        }

        //read
        try (BufferedReader reader = new BufferedReader(new
FileReader(filename))) {
            String line;
            int row = 0;

            while ((line = reader.readLine()) != null) {
                String[] values = line.trim().split("\\s+");
                for (int col = 0; col < values.length; col++) {
                    matrix[row][col] =
Integer.parseInt(values[col]);
                }
                row++;
            }
            return matrix;
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage());
        }
    }
```

```
    //helping method to write and read a file for vector elements
    public static int[] vectorCreate(int N, boolean generateRandom,
Scanner scan, int inputNumberParam) {
        String filename = scan.nextLine();
        int[] vector = new int[N];

        //write
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filename))) {
            for (int i = 0; i < vector.length; i++) {
                int value = generateRandom ? new
Random().nextInt(10) : inputNumberParam;
                writer.write(value + " ");
            }
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage());
        }

        //read
        try (BufferedReader reader = new BufferedReader(new
FileReader(filename))) {
            String line = reader.readLine();
            if (line != null) {
                String[] values = line.trim().split("\\s+");
                vector = new int[values.length];
                for (int i = 0; i < values.length; i++) {
                    vector[i] = Integer.parseInt(values[i]);
                }
                return vector;
            }
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage());
        }
        return vector;
    }

    //helping method to write a file for the final result (both
vector and matrix)
    public static <T> void writeResultToFile(T res, Scanner scan) {
        String filename = scan.nextLine();
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filename))) {
            if (res instanceof int[]) {
                for (int value : (int[]) res) {
                    writer.write(value + " ");
                }
            } else if (res instanceof int[][]) {
```

```
                for (int[] row : (int[][]) res) {
                    for (int value : row) {
                        writer.write(value + " ");
                    }
                    writer.newLine();
                }
            } else {
                throw new IllegalArgumentException("Unsupported
data type");
            }
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage());
        }
    }
}
```

T1_F1.java

```
import java.util.Arrays;
import java.util.Scanner;

public class T1_F1 extends Thread {
    //local variables
    int [] A, C, B, E;
    int [][] MA, ME;

    public void inputAndCalculate() {
        try (Scanner scan = new Scanner(System.in)) {
            System.out.print("Enter N: ");
            int N = scan.nextInt();

            if (N == 4) {
                //input vector values
                System.out.print("Enter A values (separated by
spaces): ");
                A = Data.vectorInput(N, scan);

                System.out.print("Enter B values (separated by
spaces): ");
                B = Data.vectorInput(N, scan);

                System.out.print("Enter C values (separated by
spaces): ");
                C = Data.vectorInput(N, scan);

                //input matrix values
                System.out.println("Enter MA values:");
```

```
                MA = Data.matrixInput(N, scan);

                System.out.println("Enter ME values:");
                ME = Data.matrixInput(N, scan);

                //final result calculation
                E = Data.F1(A, B, C, MA, ME);
            } else if (N > 1000) {
                System.out.println("Vector and matrix values are to
be read from file.");
                System.out.println("Press 1 to generate random
values or 2 for them to be equal 1: ");
                int choice = scan.nextInt();
                scan.nextLine();

                switch (choice) {
                    case 1:
                        //vector files creation
                        System.out.println("Enter file name for A
vector: ");
                        A = Data.vectorCreate(N,true, scan, 1);

                        System.out.println("Enter file name for B
vector: ");
                        B = Data.vectorCreate(N,true, scan, 1);

                        System.out.println("Enter file name for C
vector: ");
                        C = Data.vectorCreate(N,true, scan, 1);

                        //matrix files creation
                        System.out.println("Enter file name for MA
matrix: ");
                        MA = Data.matrixCreate(N, true, scan, 1);

                        System.out.println("Enter file name for ME
matrix: ");
                        ME = Data.matrixCreate(N, true, scan, 1);

                        //final result calculation and writing it to
a file
                        E = Data.F1(A, B, C, MA, ME);
                        System.out.println("Enter file name for E
(final result) vector: ");
                        Data.writeResultToFile(E, scan);
                        break;
                    case 2:
                        //vector files creation
```

```
                        System.out.println("Enter file name for A
vector: ");
                        A = Data.vectorCreate(N,false, scan, 1);

                        System.out.println("Enter file name for B
vector: ");
                        B = Data.vectorCreate(N, false, scan, 1);

                        System.out.println("Enter file name for C
vector: ");
                        C = Data.vectorCreate(N, false, scan, 1);

                        //matrix files creation
                        System.out.println("Enter file name for MA
matrix: ");
                        MA = Data.matrixCreate(N, false, scan, 1);

                        System.out.println("Enter file name for ME
matrix: ");
                        ME = Data.matrixCreate(N,false, scan, 1);

                        //final result calculation and writing it to
a file
                        E = Data.F1(A, B, C, MA, ME);
                        System.out.println("Enter file name for E
(final result) vector: ");
                        Data.writeResultToFile(E, scan);
                        break;
                    default:
                        System.out.println("Invalid choice. Please
select 1 or 2.");
                }
            }
        } catch (Exception e) {
            System.out.println("ERROR: " + e.getMessage());
        }
    }

    @Override
    public void run(){
        System.out.println("T1 is started");
        inputAndCalculate();
        System.out.println("E = " + Arrays.toString(E));
        System.out.println("T1 is finished");
    }
}
```

T2_F2.java

```
import java.util.Arrays;
import java.util.Scanner;

public class T2_F2 extends Thread{
   int[][] MK, ML, MG, MF;

   public void inputAndCalculate() {
       try (Scanner scan = new Scanner(System.in)) {
           System.out.print("Enter N: ");
           int N = scan.nextInt();

           if (N == 4) {
               //input matrix values
               System.out.println("Enter MK values:");
               MK = Data.matrixInput(N, scan);

               System.out.println("Enter ML values:");
               ML = Data.matrixInput(N, scan);

               System.out.println("Enter MG values:");
               MG = Data.matrixInput(N, scan);

               //final result calculation
               MF = Data.F2(MK, ML, MG);
           } else if (N > 1000) {
               System.out.println("Matrix values are to be read
from file.");
               System.out.println("Press 1 to generate random
values or 2 for them to be equal 2: ");
               int choice = scan.nextInt();
               scan.nextLine();

               switch (choice) {
                   case 1:
                       //matrix files creation
                       System.out.println("Enter file name for MK
matrix: ");
                       MK = Data.matrixCreate(N, true, scan, 2);

                       System.out.println("Enter file name for ML
matrix: ");
                       ML = Data.matrixCreate(N, true, scan, 2);

                       System.out.println("Enter file name for MG
matrix: ");
                       MG = Data.matrixCreate(N, true, scan, 2);
```

```
                        //final result calculation and writing it to
a file
                        MF = Data.F2(MK, ML, MG);
                        System.out.println("Enter file name for MF
(final result) matrix: ");
                        Data.writeResultToFile(MF, scan);
                        break;
                    case 2:
                        //matrix files creation
                        System.out.println("Enter file name for MK
matrix: ");
                        MK = Data.matrixCreate(N, false, scan, 2);

                        System.out.println("Enter file name for ML
matrix: ");
                        ML = Data.matrixCreate(N, false, scan, 2);

                        System.out.println("Enter file name for MG
matrix: ");
                        MG = Data.matrixCreate(N, false, scan, 2);

                        //final result calculation and writing it to
a file
                        MF = Data.F2(MK, ML, MG);
                        System.out.println("Enter file name for MF
(final result) matrix: ");
                        Data.writeResultToFile(MF, scan);
                        break;
                    default:
                        System.out.println("Invalid choice. Please
select 1 or 2.");
                }
            }
        } catch (Exception e) {
            System.out.println("ERROR: " + e.getMessage());
        }
    }

    @Override
    public void run(){
        System.out.println("T2 is started");
        inputAndCalculate();
        System.out.println("MF = " + Arrays.deepToString(MF));
        System.out.println("T2 is finished");
    }
}
```

T3_F3.java

```
import java.util.Arrays;
import java.util.Scanner;

public class T3_F3 extends Thread {
   int[] P, O;
   int[][] MR, MS;

   public void inputAndCalculate() {
       try (Scanner scan = new Scanner(System.in)) {
           System.out.print("Enter N: ");
           int N = scan.nextInt();

           if (N == 4) {
               //input vector values
               System.out.print("Enter P values (seperated by
spaces): ");
               P = Data.vectorInput(N, scan);

               //input matrix values
               System.out.println("Enter MR values: ");
               MR = Data.matrixInput(N, scan);

               System.out.println("Enter MS values: ");
               MS = Data.matrixInput(N, scan);

               //final result calculation
               O = Data.F3(P, MR, MS);
           } else if (N > 1000) {
               System.out.println("Vector and matrix values are to
be read from file.");
               System.out.println("Press 1 to generate random
values or 2 for them to be equal 3: ");
               int choice = scan.nextInt();
               scan.nextLine();

               switch (choice) {
                   case 1:
                       //vector files creation
                       System.out.println("Enter file name for P
vector: ");
                       P = Data.vectorCreate(N, true, scan, 3);

                       //matrix files creation
                       System.out.println("Enter file name for MR
matrix: ");
```

```
                        MR = Data.matrixCreate(N, true, scan, 3);

                        System.out.println("Enter file name for MS
matrix: ");
                        MS = Data.matrixCreate(N, true, scan, 3);

                        //final result calculation and writing it to
a file
                       O = Data.F3(P, MR, MS);
                        System.out.println("Enter file name for O
(final result) vector: ");
                        Data.writeResultToFile(O, scan);
                        break;
                    case 2:
                        //vector files creation
                       System.out.println("Enter file name for P
vector: ");
                        P = Data.vectorCreate(N, false, scan, 3);

                        //matrix files creation
                       System.out.println("Enter file name for MR
matrix: ");
                        MR = Data.matrixCreate(N, false, scan, 3);

                        System.out.println("Enter file name for MS
matrix: ");
                        MS = Data.matrixCreate(N, false, scan, 3);

                        //final result calculation and writing it to
a file
                       O = Data.F3(P, MR, MS);
                        System.out.println("Enter file name for O
(final result) vector: ");
                        Data.writeResultToFile(O, scan);
                        break;
                    default:
                        System.out.println("Invalid choice. Please
select 1 or 2.");
                }
            }
        } catch (Exception e) {
            System.out.println("ERROR: " + e.getMessage());
        }
    }

    @Override
    public void run(){
        System.out.println("T3 is started");
```

```
        inputAndCalculate();
        System.out.println("O = " + Arrays.toString(O));
        System.out.println("T3 is finished");
    }
}
```

Висновки:

Було виконано розробку паралельної багатопотокової програми на мові **Java** за допомогою екземплярів класів, що в свою чергу є нащадками класу **Thread**. За особливістю мови кожен клас потоку мав свій перевизначений метод `run()`, що описував послідовність кроків при його виконанні. Також був створений головний клас **Lab1**, який відповідає за запуск усіх трьох потоків та використовує метод `join()` для синхронізації свого виконання та виконання інших потоків. Окрім цього в окремому класі **Data** були визначені допоміжні методи для обрахунку, вводу й виводу даних, запису та читання файлів.

При виконанні роботи виникла проблема із введенням потрібних даних. Так як потоки є незалежними один від одного, запит на отримання першочергового значення N відбувався одночасно. Використання одного засобу вводу даних (консолі) призводило до невпорядкованого встановлення значень для потрібних змінних. Як результат відбувались колізії у обрахунках та потоки завершували своє виконання без успішного фінального результату. Причиною слугує невпорядкованість потоків між собою. Тому розв'язком даної проблеми може бути використання спеціальних засобів, таких як семафори, м'ютекси або інші модифікатори для контролю доступу до змінних.

Також було проведено дослідження завантаженості ядер процесора при виконанні багатопотокової програми. На отриманих графіках видно, що при існуючій можливості, операційна система намагається якомога оптимальніше розподілити навантаження. Якщо ж обмежити використання ядер, зменшивши їх кількість у Диспетчері задач, то навантаження на кожне залишене ядро значно зросте, про що сигналізують скачки у відповідних графіках. Окрім цього був обрахований повний час виконання потоку T2_F2 при використанні 4 ядер та 1 ядра. Коефіцієнт прискорення: Ку = 32344 / 20969 = 1.542. Це також доводить, що при використанні меншої кількості ядер процесора, навантаження на кожне з них зростає і швидкість виконання програми спадає.