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 №4

«Програмування для комп'ютерних систем зі спільною пам'яттю. Монітори»

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: розробка програми для ПКС зі СП
Мова програмування: Java
Засоби організації взаємодії процесів: монітори мови Java

Вхідні дані:

- комп'ютерна система зі спільною пам'яттю включає чотири процесори і чотири пристрої введення-виведення (рис. 1.1);
- математичне завдання згідно **варіанту №8**:
  X = p * max(C * (MA * MD)) * R + e * B,
  де X, B, R, C - вектори розміру N; MA, MD - матриці розміру N; p, e - скаляри;
- введення значення e та виведення результату X виконується у процесорі 1;
  введення значення для C, MA виконується у процесорі 2;
  введення значень для R, MD виконується у процесорі 3;
  введення значень для B, p виконується у процесорі 4.

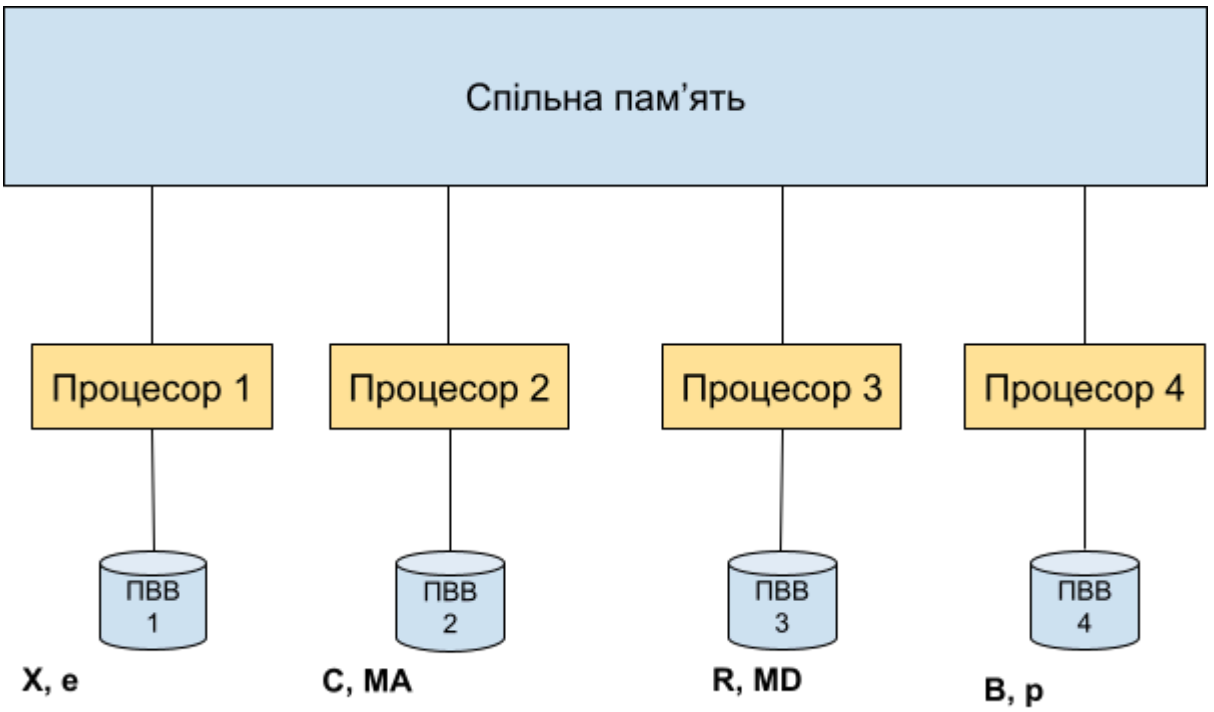


рис. 1.1 Структурна схема 4 процесорної системи

ВИКОНАННЯ РОБОТИ

Етап 1. Побудова паралельного алгоритму

$$X = p * \max(C * (MA * MD)) * R + e * B$$

1. $ai = \max(C * (MA * MDн)),\ i = 1...P$
2. $a = \max(a, ai)$ CP: a
3. $Xн = p * a * Rн + e * Bн$ CP: p, e, a - копії

,де н = N/P частина вектору або матриці, N - розмір вектору/матриці, P = 4 - кількість процесорів (потоків).

Етап 2. Розробка алгоритмів потоків

Задача T1	**Точки синхронізації, КД**
1. Введення e	
2. Сигнал задачі T2, T3, T4 про введення e	S2-1 S3-1 S4-1
3. Чекати на введення C, MA у задачі T2; R, MD у задачі T3; B, p у задачі T4	W2-1 W3-1 W4-1
4. Обчислення1 a1 = max(C * (MA * MDн))	
5. Обчислення2 a = max(a, a1)	КД1
6. Сигнал T2, T3, T4 про завершення обчислень a	S2-2 S3-2 S4-2
7. Чекати на завершення обчислень a в T2, T3, T4	W2-2 W3-2 W4-2
8. Копія a1 = a	КД2
9. Копія p1 = p	КД3
10. Копія e1 = e	КД4
11. Обчислення3 Xн = p1 * a1 * Rн + e1 * Bн	

12. Чекати на завершення обчислень Xн в T2, T3, T4	W2-3 W3-3 W4-3
13. Вивід X	

Задача T2	**Точки синхронізації, КД**
1. Введення C, MA	
2. Сигнал задачі T1, T3, T4 про введення C, MA	S1-1 S3-1 S4-1
3. Чекати на введення e у задачі T1; R, MD у задачі T3; B, p у задачі T4	W1-1 W3-1 W4-1
4. Обчислення1 a2 = max(C * (MA * MDн))	
5. Обчислення2 a = max(a, a2)	КД1
6. Сигнал T1, T3, T4 про завершення обчислень a	S1-2 S3-2 S4-2
7. Чекати на завершення обчислень a в T1, T3, T4	W1-2 W3-2 W4-2
8. Копія a2 = a	КД2
9. Копія p2 = p	КД3
10. Копія e2 = e	КД4
11. Обчислення3 Xн = p2 * a2 * Rн + e2 * Bн	
12. Сигнал T1 про завершення обчислень Xн	S1-3

Задача T3	**Точки синхронізації, КД**
1. Введення R, MD	

2. Сигнал задачі T1, T2, T4 про введення R, MD	S1-1 S2-1 S4-1
3. Чекати на введення e у задачі T1; C, MA у задачі T2; B, p у задачі T4	W1-1 W2-1 W4-1
4. Обчислення1 a3 = max(C * (MA * MDн))	
5. Обчислення2 a = max(a, a3)	КД1
6. Сигнал T1, T2, T4 про завершення обчислень a	S1-2 S2-2 S4-2
7. Чекати на завершення обчислень a в T1, T2, T4	W1-2 W2-2 W4-2
8. Копія a3 = a	КД2
9. Копія p3 = p	КД3
10. Копія e3 = e	КД4
12. Обчислення3 Xн = p3 * a3 * Rн + e3 * Bн	
12. Сигнал T1 про завершення обчислень Xн	S1-3

Задача T4	**Точки синхронізації, КД**
1. Введення B, p	
2. Сигнал задачі T1, T2, T3 про введення B, p	S1-1 S2-1 S3-1
3. Чекати на введення e у задачі T1; C, MA у задачі T2; R, MD у задачі T3	W1-1 W2-1 W3-1
4. Обчислення1 a4 = max(C * (MA * MDн))	
5. Обчислення2 a = max(a, a4)	КД1
6. Сигнал T1, T2, T3 про завершення обчислень a	S1-2 S2-2 S3-2

7. Чекати на завершення обчислень a в T1, T2, T3	W1-2 W2-2 W3-2
8. Копія a4 = a	КД2
9. Копія p4 = p	КД3
11. Копія e4 = e	КД4
12. Обчислення4 Xн = p4 * a4 * Rн + e4 * Bн	
13. Сигнал T1 про завершення обчислень Xн	S1-3

Етап 3. Розробка схеми взаємодії задач

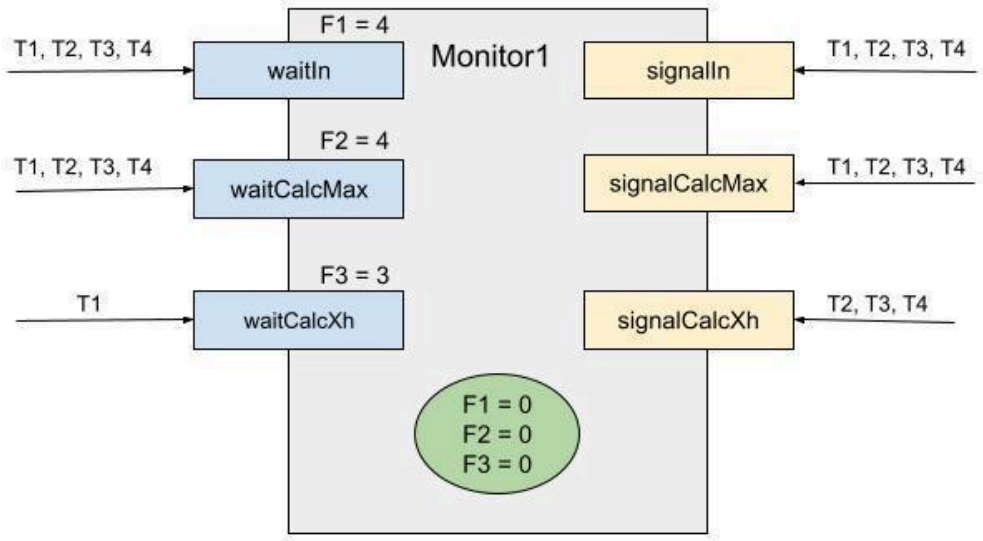


Рис.1.2 Схема взаємодії потоків через клас-монітор Monitor1

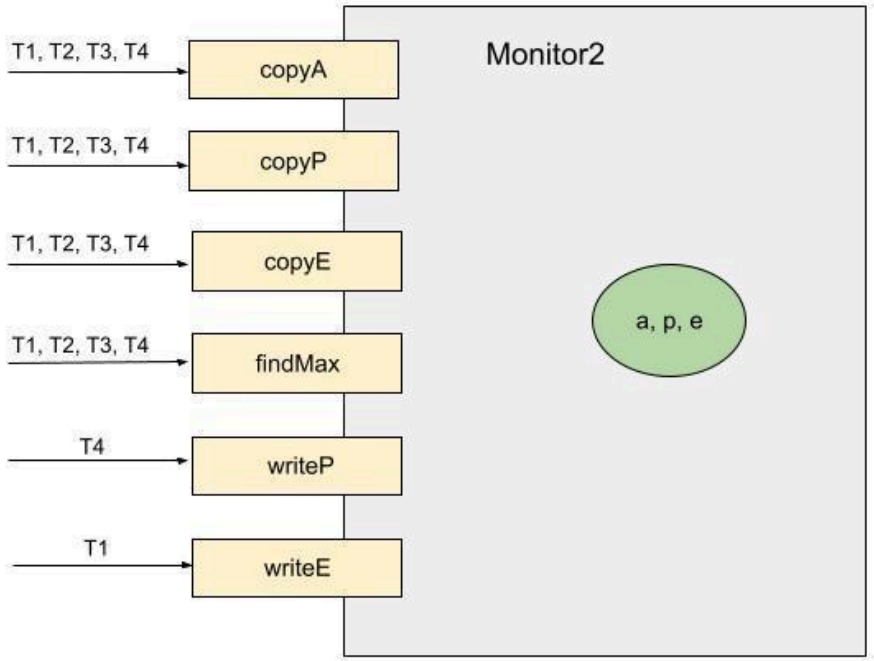


Рис.1.3 Схема взаємодії потоків через клас-монітор Monitor2

Захищений модуль **Monitor1** (рис.1.2) призначений для синхронізації взаємодії потоків; включає три захищені елементи: F1, F2 і F3, а також самостійно розроблених методів захищених операцій

- **signalIn -** для сигналу про завершення введення даних в задачах T1, T2, T3, T4;
- **waitIn -** для очікування завершення введення даних в задачах T1, T2, T3, T4;
- **signalCalcMax -** для сигналу про завершення обчислення max(C * (MA * MDн)) в задачах T1, T2, T3, T4;
- **waitCalcMax -** для очікування завершення обчислення max(C * (MA * MDн)) в задачах T1, T2, T3, T4;
- **signalCalcXh -** для сигналу про завершення обчислення Xн в задачах T1, T2, T3, T4;
- **waitCalcXh -** для очікування завершення обчислення Xн в задачах T1, T2, T3, T4;

Захищений модуль **Monitor2** (рис.1.3) приначений для вирішення задач взаємного виключення; включає три захищені елементи: a, p, e, а також набір захищених операцій:

- **findMax -** для визначення максимального значення змінної a;
- **writeP -** для задання значення змінної p;
- **writeE -** для задання значення змінної e;
- **copyA -** для копіювання значення змінної a;
- **copyP -** для копіювання значення змінної p;
- **copyE -** для копіювання значення змінної e;

Етап 4. Розробка програми

Програма складається з основного класу Lab4, допоміжного класу з методами для обчислення математичних виразів Data, класів-моніторів Monitor1 і Monitor2 та 4 класів для кожного з 4 потоків: T1, T2, T3, T4.

Lab4.java

```
//Дисципліна "Програмне забезпечення високопродуктивних комп'ютерних систем"
//Лабораторна робота ЛР4 Варіант 8
//X = p * max(C * (MA * MD)) * R + e * B
//Мартинюк Марія Павлівна
//Дата 26.04.2024

public class Lab4 {
   public static void main(String []args) {
        double start, end, totalTime;
        start = System.currentTimeMillis();

        Data data = new Data();

        T1 T1 = new T1(data);
        T2 T2 = new T2(data);
        T3 T3 = new T3(data);
        T4 T4 = new T4(data);

        //threads start
        T1.start();
        T2.start();
        T3.start();
        T4.start();

        try {
            T1.join();
            T2.join();
            T3.join();
```

```
            T4.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;

public class Data {
    static int N = 8;
    static int P = 4;
    static int H = N/P;

    public Monitor1 monitor1 = new Monitor1();
    public Monitor2 monitor2 = new Monitor2();

    static int[] C, R, B;
    static int[][] MA, MD;

    //допоміжний метод для заповнення вектору одиницями
    public static int[] vectorInput(int N) {
        int[] vector = new int[N];
        Arrays.fill(vector, 1);
        return vector;
    }

    //допоміжний метод для заповнення матриці одиницями
    public static int[][] matrixInput(int N) {
        int[][] matrix = new int[N][N];
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                matrix[i][j] = 1;
            }
        }
        return matrix;
    }

    //допоміжний метод для розрахунку добутку матриць
    public static int[][] multiplyMatricesPartialResult(int[][]
matrix1, int[][] matrix2, int startRow, int endRow) {
```

```
        int[][] partialResult = new int[endRow -
startRow][matrix2[0].length];
        for (int i = startRow; i < endRow; i++) {
            for (int j = 0; j < matrix2[0].length; j++) {
                for (int k = 0; k < matrix2.length; k++) {
                    partialResult[i - startRow][j] += matrix1[i][k] *
matrix2[k][j];
                }
            }
        }
        return partialResult;
    }

    //допоміжний метод для розрахунку добутку матриці на вектор
    public static int[] multiplyMatrixVector(int[][] matrix, int[]
vector) {
        int numRows = matrix.length;
        int numCols = matrix[0].length;
        int[] result = new int[numRows];

        for (int i = 0; i < numRows; i++) {
            int sum = 0;
            for (int j = 0; j < numCols; j++) {
                sum += matrix[i][j] * vector[j];
            }
            result[i] = sum;
        }
        return result;
    }

    //допоміжний метод для пошуку максимального значення у векторі
     public static int findMaxValue(int[] vector) {
        int max = vector[0];

        for (int i = 1; i < vector.length; i++) {
            if (vector[i] > max) {
                max = vector[i];
            }
        }
        return max;
     }

    //допоміжний метод для розрахунку добутку вектора на скаляр
    public static int[] multiplyVectorScalar(int[] vector, int
scalar) {
        int length = vector.length;
        int[] result = new int[length];
```

```
        for (int i = 0; i < length; i++) {
            result[i] = vector[i] * scalar;
        }
        return result;
    }

    //допоміжний метод для розрахунку суми векторів
    public static int[] addVectors(int[] vector1, int[] vector2) {
        int length = vector1.length;
        int[] result = new int[length];

        for (int i = 0; i < length; i++) {
            result[i] = vector1[i] + vector2[i];
        }
        return result;
    }

    //допоміжний метод для розрахунку Xн = pi * ai * Rн + ei * Bн
    public static int[] calculateXh(int pi, int ai, int ei, int[] R,
int[] B) {
        int x = pi * ai;
        int[] resPart1 = multiplyVectorScalar(R, x);
        int[] resPart2 = multiplyVectorScalar(B, ei);
        int[] Xh = addVectors(resPart1, resPart2);
        return Xh;
    }

    //допоміжний метод для об'єднання та виводу результуючого вектора
    public static void mergeAndPrintX() {
        int totalLength = T1.Xh.length + T2.Xh.length + T3.Xh.length
+ T4.Xh.length;
        int[] X = new int[totalLength];
        System.arraycopy(T1.Xh, 0, X, 0, T1.Xh.length);
        System.arraycopy(T2.Xh, 0, X, T1.Xh.length, T2.Xh.length);
        System.arraycopy(T3.Xh, 0, X, T1.Xh.length + T2.Xh.length,
T3.Xh.length);
        System.arraycopy(T4.Xh, 0, X, T1.Xh.length + T2.Xh.length +
T3.Xh.length, T4.Xh.length);
        System.out.println("X = " + Arrays.toString(X));
    }
}
```

Monitor1.java

```
public class Monitor1 {
    private int F1 = 0;
    private int F2 = 0;
    private int F3 = 0;
```

```
public synchronized void signalIn() {
    F1 += 1;
    if (F1 == 4) {
        notifyAll();
    }
}

public synchronized void waitIn() {
    try {
        if (F1 != 4) {
            wait();
        }
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
}

public synchronized void signalCalcMax() {
    F2 += 1;
    if (F2 == 4) {
        notifyAll();
    }
}

public synchronized void waitCalcMax() {
    try {
        if (F2 != 4) {
            wait();
        }
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
}

public synchronized void signalCalcXh() {
    F3 += 1;
    if (F3 == 3) {
        notify();
    }
}

public synchronized void waitCalcXh() {
    try {
        if (F3 != 3) {
            wait();
        }
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
```

```
        }
    }
}
```

Monitor2.java

```
public class Monitor2 {
    private int a, p, e;

    public synchronized void findMax(int ai) {
        this.a = Math.max(a, ai);
    }

    public synchronized void writeP(int value) {
        this.p = value;
    }

    public synchronized void writeE(int value) {
        this.e = value;
    }

    public synchronized int copyA() {
        return this.a;
    }

    public synchronized int copyP() {
        return this.p;
    }

    public synchronized int copyE() {
        return this.e;
    }
}
```

T1.java

```
public class T1 extends Thread {
    private Data data;
    private int a1, p1, e1;
    private int threadIndex = 1;
    static int[] Xh;

    public T1(Data data) {
        this.data = data;
    }

    public void inputAndCalculate() {
        // Введення e
        data.monitor2.writeE(1);
```

```
        //Сигнал задачі T2, T3, T4 про введення e
        data.monitor1.signalIn();

        // Чекати на введення C, MA у задачі T2; R, MD у задачі T3;
B, p у задачі T4
        data.monitor1.waitIn();

        // Обчислення1 a1 = max(C * (MA * MDн))
        int startIndex = (threadIndex * data.H) - data.H;
        int[][] matrixMultiply =
data.multiplyMatricesPartialResult(data.MA, data.MD, startIndex,
startIndex + data.H);
        int[] matrixVectorMultiply =
data.multiplyMatrixVector(matrixMultiply, data.C);
        a1 = data.findMaxValue(matrixVectorMultiply);

        // Обчислення2 a = max(a, a1)
        data.monitor2.findMax(a1); // КД1

        // Сигнал T2, T3, T4 про завершення обчислень a
        data.monitor1.signalCalcMax();

        // Чекати на завершення обчислень a в T2, T3, T4
        data.monitor1.waitCalcMax();

        // Копія a1 = a
        a1 = data.monitor2.copyA(); // КД2

        // Копія p1 = p
        p1 = data.monitor2.copyP(); // КД3

        // Копія e1 = e
        e1 = data.monitor2.copyE(); // КД4

        // Обчислення3 Xн = p1 * a1 * Rн + e1 * Bн
        int[] Rh = new int[data.H];
        System.arraycopy(data.R, startIndex, Rh, 0, data.H);
        int[] Bh = new int[data.H];
        System.arraycopy(data.B, startIndex, Bh, 0, data.H);
        Xh = data.calculateXh(p1, a1, e1, Rh, Bh);

        // Чекати на завершення обчислень Xн в T2, T3, T4
        data.monitor1.waitCalcXh();

        // Вивід X
        data.mergeAndPrintX();

    }
```

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

T2.java

```
import java.util.Arrays;

public class T2 extends Thread {
    private Data data;
    private int a2, p2, e2;
    private int threadIndex = 2;
    static int[] Xh;

    public T2(Data data) {
        this.data = data;
    }

    public void inputAndCalculate() {
        // Введення C, MA
        data.C = data.vectorInput(data.N);
        data.MA = data.matrixInput(data.N);

        // Сигнал задачі T1, T3, T4 про введення C, MA
        data.monitor1.signalIn();

        // Чекати на введення e у задачі T1; R, MD у задачі T3; B, p
у задачі T4
        data.monitor1.waitIn();

        // Обчислення1 a2 = max(C * (MA * MDн))
        int startIndex = (threadIndex * data.H) - data.H;
        int[][] matrixMultiply =
data.multiplyMatricesPartialResult(data.MA, data.MD, startIndex,
startIndex + data.H);
        int[] matrixVectorMultiply =
data.multiplyMatrixVector(matrixMultiply, data.C);

        matrixVectorMultiply[0] = 2; // Встановлення відмінного від
інших значення елемента для перевірки функції max
        a2 = data.findMaxValue(matrixVectorMultiply);

        System.out.println("C * (MA * MDн) = " +
Arrays.toString(matrixVectorMultiply));
```

```
        System.out.println("a2 = " + a2);

        // Обчислення2 a = max(a, a2)
        data.monitor2.findMax(a2); // КД1

        // Сигнал T1, T3, T4 про завершення обчислень a
        data.monitor1.signalCalcMax();

        // Чекати на завершення обчислень a в T1, T3, T4
        data.monitor1.waitCalcMax();

        // Копія a2 = a
        a2 = data.monitor2.copyA(); // КД2

        // Копія p2 = p
        p2 = data.monitor2.copyP(); // КД3

        // Копія e2 = e
        e2 = data.monitor2.copyE(); // КД4

        // Обчислення3 Xh = p2 * a2 * Rh + e2 * Bh
        int[] Rh = new int[data.H];
        System.arraycopy(data.R, startIndex, Rh, 0, data.H);
        int[] Bh = new int[data.H];
        System.arraycopy(data.B, startIndex, Bh, 0, data.H);
        Xh = data.calculateXh(p2, a2, e2, Rh, Bh);

        // Сигнал T1 про завершення обчислень Xh
        data.monitor1.signalCalcXh();

    }

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

T3.java

```
public class T3 extends Thread {
    private Data data;
    private int a3, p3, e3;
    private int threadIndex = 3;
    static int[] Xh;

    public T3(Data data) {
```

```
        this.data = data;
    }

    public void inputAndCalculate() {
        // Введення R, MD
        data.R = data.vectorInput(data.N);
        data.MD = data.matrixInput(data.N);

        // Сигнал задачі T1, T2, T4 про введення R, MD
        data.monitor1.signalIn();

        // Чекати на введення e у задачі T1; C, MA у задачі T2; B, p
у задачі T4
        data.monitor1.waitIn();

        // Обчислення1 a3 = max(C * (MA * MDн))
        int startIndex = (threadIndex * data.H) - data.H;
        int[][] matrixMultiply =
data.multiplyMatricesPartialResult(data.MA, data.MD, startIndex,
startIndex + data.H);
        int[] matrixVectorMultiply =
data.multiplyMatrixVector(matrixMultiply, data.C);
        a3 = data.findMaxValue(matrixVectorMultiply);

        // Обчислення2 a = max(a, a3)
        data.monitor2.findMax(a3); // КД1

        // Сигнал T1, T2, T4 про завершення обчислень a
        data.monitor1.signalCalcMax();

        // Чекати на завершення обчислень a в T1, T2, T4
        data.monitor1.waitCalcMax();

        // Копія a3 = a
        a3 = data.monitor2.copyA(); // КД2

        // Копія p3 = p
        p3 = data.monitor2.copyP(); // КД3

        // Копія e3 = e
        e3 = data.monitor2.copyE(); // КД4

        // Обчислення3 Xн = p3 * a3 * Rн + e3 * Bн
        int[] Rh = new int[data.H];
        System.arraycopy(data.R, startIndex, Rh, 0, data.H);
        int[] Bh = new int[data.H];
        System.arraycopy(data.B, startIndex, Bh, 0, data.H);
        Xh = data.calculateXh(p3, a3, e3, Rh, Bh);
```

```
        // Сигнал T1 про завершення обчислень Xh
        data.monitor1.signalCalcXh();
    }

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

T4.java

```
public class T4 extends Thread {
    private Data data;
    private int a4, p4, e4;
    private int threadIndex = 4;
    static int[] Xh;

    public T4(Data data) {
        this.data = data;
    }

    public void inputAndCalculate() {
        // Введення B, p
        data.B = data.vectorInput(data.N);
        data.monitor2.writeP(1);

        // Сигнал задачі T1, T2, T3 про введення B, p
        data.monitor1.signalIn();

        // Чекати на введення e у задачі T1; C, MA у задачі T2; R, MD
у задачі T3
        data.monitor1.waitIn();

        // Обчислення1 a4 = max(C * (MA * MDн))
        int startIndex = (threadIndex * data.H) - data.H;
        int[][] matrixMultiply =
data.multiplyMatricesPartialResult(data.MA, data.MD, startIndex,
startIndex + data.H);
        int[] matrixVectorMultiply =
data.multiplyMatrixVector(matrixMultiply, data.C);
        a4 = data.findMaxValue(matrixVectorMultiply);

        // Обчислення2 a = a + a4
        data.monitor2.findMax(a4); // КД1
```

```
        // Сигнал T1, T2, T3 про завершення обчислень a
        data.monitor1.signalCalcMax();

        // Чекати на завершення обчислень a в T1, T2, T3
        data.monitor1.waitCalcMax();

        // Копія a4 = a
        a4 = data.monitor2.copyA(); // КД2

        // Копія p4 = p
        p4 = data.monitor2.copyP(); // КД3

        // Копія e4 = e
        e4 = data.monitor2.copyE(); // КД4

        // Обчислення3 Xн = p4 * a4 * Rн + e4 * Bн
        int[] Rh = new int[data.H];
        System.arraycopy(data.R, startIndex, Rh, 0, data.H);
        int[] Bh = new int[data.H];
        System.arraycopy(data.B, startIndex, Bh, 0, data.H);
        Xh = data.calculateXh(p4, a4, e4, Rh, Bh);

        // Сигнал T1 про завершення обчислень Xн
        data.monitor1.signalCalcXh();
    }

    @Override
    public void run(){
        System.out.println("T4 is started");
        inputAndCalculate();
        System.out.println("T4 is finished");
    }
}
```

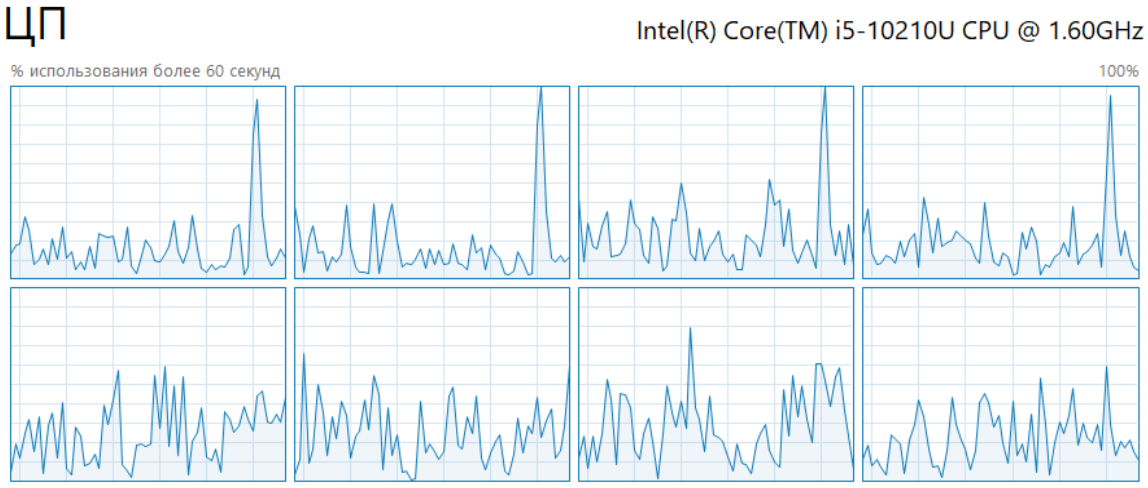
Результат виконання:

На рис. нижче наведений результат виконання багатопотокової програми при N = 8, P = 4. Значення усіх елементів використовуваних матриць та векторів рівні 1. Задля впевненості у правильності розрахунків функції max було додатково змінено значення одного елементу у результуючого вектора (C * (MA * MDн)) у потоці T2. Також виведено у консоль проміжні результати обчислень, що підтверджують правильність визначення максимального значення елементів. Як видно з рис. нижче разрахунок частин та об’єднання їх у остаточний результат відбувається коректно.

```
T1 is started
T4 is started
T2 is started
T3 is started
C * (MA * MDн) = [2, 64]
a2 = 64
T2 is finished
T3 is finished
T4 is finished
X = [65, 65, 65, 65, 65, 65, 65, 65]
T1 is finished
TIME 15.0ms
```

Дослідження завантаженості ядер процесору:

Варто зазначити, що конкретно мій комп'ютер містить 8 логічних ядер та 4 фізичних. Нижче наведений рис. демонструє, що при використанні 4 ядер процесору розподілення обчислювальних можливостей відбувається рівномірно між кожним з них. Поєднуючи цю інформацію із раніше отриманими остаточними значеннями обрахунків, можна зробити висновок, що чотирьох потокова програма написана коректно.



Додатково було обраховано коефіцієнт прискорення при запуску програми на 4 ядрах та на 1 ядрі при N = 8: Кп = 31 / 10 = 3,1. Тобто при використанні 4 ядер замість 1, програма завершується у 3,1 рази швидше.

Висновок:

1 Побудовано та розроблено паралельний алгоритм для обчислення математичної задачі. За отриманим алгоритмом було визначено критичні ділянки з використання спільних ресурсів a, p, e, a.

2 Задля реалізації коректної синхронізації паралельних обчислень були розроблені класи-монітори мови Java, для імплементації яких використовувався модифікатор private у спільного ресурсу; synchronized методи; wait(), notify(), notifyAll() - для рішення завдання синхронізації (блокування і розблокування потоків).

3 Побудовано схему взаємодії задач через клас-монітор Monitor1 та Monitor2, завдяки наочності якої було досягнуто успішної реалізації синхронізації паралельних обчислень.

4 Отриманий вивід результату обчислень демонструє, що розрахунок остаточного значення відбувається коректно. Була проведена додаткова перевірка на правильність розрахунку функції max зміною значення елемента вектора, що також підтвердила правильність обчислень. Аналіз завантаженості ядер процесору при запуску програми ствердив рівномірне розподілення навантаження між задіяними ядрами, що свідчить про успішну оптимізацію паралельних обчислень.

5 Обчислений коефіцієнт прискорення Кп = 3,1, що також сповіщає про пришвидшення виконання програми у разі використання 4 ядер процесора замість 1.