**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

Лабораторна робота №2

з дисципліни

«Програмне забезпечення

високопродуктивних комп'ютерних систем»

Виконав Перевірив:

Студент групи ІМ-13 доц. Корочкін О.В.

Первєєв Євгеній Олексійович

Київ 2024

Завдання на роботу

Розробити паралельну програму для обчислення в паралельної

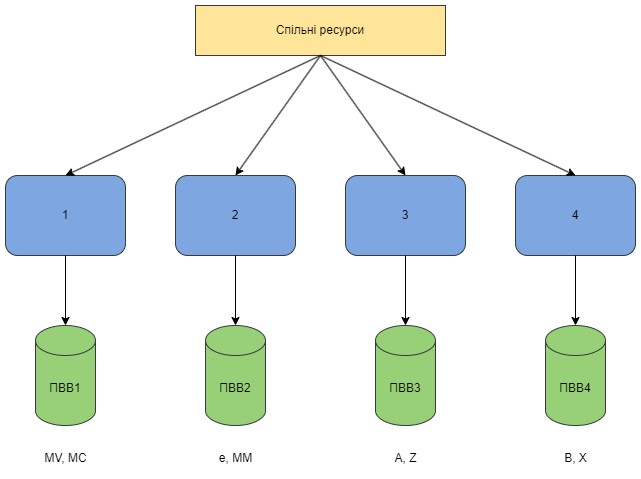
системі (ПКС СП) функції:

**Варіант:** 9

A= min(Z)\*(B\*MV) +e\*X\*(MM\*MC)

Мова програмування: Java

Засоби організації взаємодії процесів: монітори.



Етап 1. Побудова паралельного математичного алгоритму

Варіант: 9



A= min(Z)\*(B\*MV) +e\*X\*(MM\*MC)

1. ai = min(Zн)

2. a = min(a, ai) СР: а

3. Aн = a \* B \* MVн + e \* X \* (MM \* MCн) СР: B, X, MM, а, e

N - кількість елементів в вектора/матриці

P - кількість задач, які виконують обчислення

H - N / P

Етап 2. Розробка алгоритмів потоків

**Задача Т1**

1. Введення MV, MC

2. Сигнал задачам T2, T3, T4 про введення MV, MC S2-1,3-1,4-1

3. Чекати на введення даних в задачах T2, T3, T4 W2-1,3-1,4-1

4. Обчислення 1: а1 = min(Zн)

5. Обчислення 2: а = min(a, a1) КД1

6. Сигнал T2, T3, T4 про завершення обчислення а S2-2,3-2,4-2

7. Чекати на завершення обчислень а в T2, T3, T4 W2-2,3-2,4-2

8. Копіювання 1: а1 = а КД2

9. Копіювання 2: е1 = е КД3

10. Обчислення 3: Aн = a1 \* B \* MVн + e1 \* X \* (MM \* MCн)

11. Сигнал T3 про завершення обчислення Aн S3-3

**Задача Т2**

1. Введення e, MM

2. Сигнал задачам T1, T3, T4 про введення e, MM S1-1,3-1,4-1

3. Чекати на введення даних в задачах T1, T3, T4 W1-1,3-1,4-1

4. Обчислення 1: а2 = min(Zн)

5. Обчислення 2: а = min(a, a2) КД1

6. Сигнал T1, T3, T4 про завершення обчислення а S1-2,3-2,4-2

7. Чекати на завершення обчислень а в T1, T3, T4 W1-2,3-2, 4-2

8. Копіювання 1: а2 = а КД2

9. Копіювання 2: е2 = е КД3

10. Обчислення 3: Aн = a2 \* B \* MVн + e2 \* X \* (MM \* MCн)

11. Сигнал T3 про завершення обчислення Aн S3-3

**Задача Т3**

1. Введення A, Z

2. Сигнал задачам T1, T2, T4 про введення A, Z S1-1,2-1,4-1

3. Чекати на введення даних в задачах T1, T2, T4 W1-1,2-1,4-1

4. Обчислення 1: а3 = min(Zн)

5. Обчислення 2: а = min(a, a3) КД1

6. Сигнал T1, T2, T4 про завершення обчислення а S1-2,2-2,4-2

7. Чекати на завершення обчислень а в T1, T2, T4 W1-2,2-2,4-2

8. Копіювання 1: а3 = а КД2

9. Копіювання 2: е3 = е КД3

10. Обчислення 3: Aн = a3 \* B \* MVн + e3 \* X \* (MM \* MCн)

11. Чекати на завершення обчислень а в T2, T3, T4 W1-3,2-3,4-2

12. Виведення А

**Задача Т4**

1. Введення е

2. Сигнал задачам T1, T2, T3 про введення B, X S1-1,2-1,3-1

3. Чекати на введення даних в задачах T1, T2, T3 W1-1,2-1,3-1

4. Обчислення 1: а4 = min(Zн)

5. Обчислення 2: а = min(a, a4) КД1

6. Сигнал T1, T2, T3 про завершення обчислення а S1-2,2-2,3-2

7. Чекати на завершення обчислень а в T1, T2, T3 W1-2,2-2, 3-2

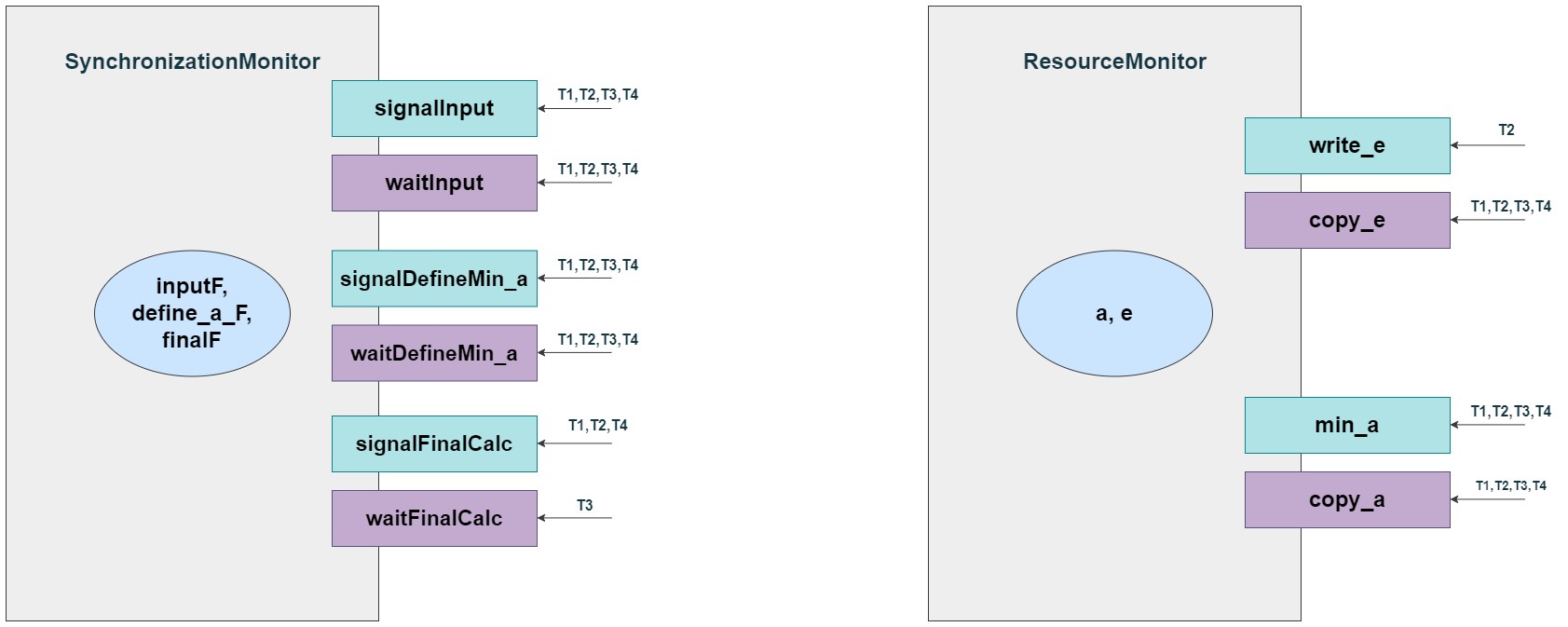
8.Копіювання 1: а4 = а КД2

9. Копіювання 2: е4 = е КД3

10. Обчислення 3: Aн = a4 \* B \* MVн + e4 \* X \* (MM \* MCн)

11. Сигнал T3 про завершення обчислення Aн S3-3

Етап 3. Розробка схеми взаємодії задач



При створенні схеми взаємодії для вирішення завдання взаємного виключення та синхронізації було використано два типи моніторів:

1. Монітор **SynchronizationMonitor** - використовується для організації сигналів та очікувань з метою синхронізації обчислень. Методи **signalInput** та **waitInput** використовуються для синхронізації введення даних. Методи **signalDefineMin\_a** та **waitDefineMin\_a**, **signalFinalCalc** та **waitFinalCalc** використовуються для синхронізації обчислення **a** та **А** відповідно.
2. Монітор **ResourceMonitor** - використовується для захисту спільних ресурсів **a** та **e**. Методи **copy\_e** та **copy\_a** використовуються для копіювання спільних ресурсів **e** та **a** відповідно. Метод **write\_e** використовується для запису значення спільного ресурсу **e**. Метод **min\_a** використовується для обчислення мінімуму спільного ресурсу **a**.

Етап 4. Розробка програми

**Main.java**

***public class*** Main {  
 ***public static void*** main(String[] args) {  
 ***long*** start = System.currentTimeMillis();  
  
 ResourceMonitor resourceMonitor = ***new*** ResourceMonitor();  
 SynchronizationMonitor synchronizationMonitor = ***new*** SynchronizationMonitor();  
  
 T1 t1 = ***new*** T1(synchronizationMonitor, resourceMonitor);  
 T2 t2 = ***new*** T2(synchronizationMonitor, resourceMonitor);  
 T3 t3 = ***new*** T3(synchronizationMonitor, resourceMonitor);  
 T4 t4 = ***new*** T4(synchronizationMonitor, resourceMonitor);  
  
 t1.start();  
 t2.start();  
 t3.start();  
 t4.start();  
  
 ***try*** {  
 t1.join();  
 t2.join();  
 t3.join();  
 t4.join();  
 } ***catch*** (InterruptedException e) {  
 ***throw new*** RuntimeException(e);  
 }  
  
 ***long*** end = System.currentTimeMillis();  
  
 System.out.printf("Execute time: %d ms", (end - start));  
 }  
}

**Data.java**

***import*** java.util.Arrays;  
  
***public class*** Data {  
 ***public final static int*** N = 16;  
 ***public final static int*** P = 4;  
 ***public final static int*** H = N / P;  
  
 ***public static int***[][] MV = ***new int***[N][N];  
 ***public static int***[][] MM = ***new int***[N][N];  
 ***public static int***[][] MC = ***new int***[N][N];  
 ***public static int***[] A = ***new int***[N];  
 ***public static int***[] Z = ***new int***[N];  
 ***public static int***[] B = ***new int***[N];  
 ***public static int***[] X = ***new int***[N];  
 ***public static int*** a;  
 ***public static int*** e;  
  
  
 ***public static int*** writeScalar() {  
 ***return*** 1;  
 }  
  
 ***public static int***[] writeVector(***int*** N) {  
 ***int***[] vector = ***new int***[N];  
 Arrays.fill(vector, 1);  
 ***return*** vector;  
 }  
  
 ***public static int***[][] writeMatrix(***int*** N) {  
 ***int***[][] matrix = ***new int***[N][N];  
 ***for***(***int*** i = 0; i < N; i++) {  
 ***for*** (***int*** t = 0; t < N; t++) {  
 matrix[i][t] = 1;  
 }  
 }  
 ***return*** matrix;  
 }  
  
 ***public static int*** findMinVector(***int***[] vector) {  
 ***int*** min = vector[0];  
 ***for***(***int*** i = 0; i < vector.length; i++) {  
 min = Math.min(vector[i], min);  
 }  
 ***return*** min;  
 }  
  
 ***public static int***[][] submatrixFromRangeOfColumns(***int***[][] matrix, ***int*** start, ***int*** end) {  
 ***int*** rows = matrix.length;  
 ***int*** columns = end - start;  
  
 ***int***[][] submatrix = ***new int***[rows][columns];  
  
 ***for*** (***int*** i = 0; i < rows; i++) {  
 ***for*** (***int*** j = start; j < end; j++) {  
 submatrix[i][j - start] = matrix[i][j];  
 }  
 }  
  
 ***return*** submatrix;  
 }  
  
 ***private static int***[] multVectorByMatrix(***int***[] vector, ***int***[][] matrix) {  
 ***int***[] res = ***new int***[matrix[0].length];  
  
 ***for*** (***int*** i = 0; i < matrix[0].length; i++) {  
 ***int*** sum = 0;  
 ***for*** (***int*** j = 0; j < vector.length; j++) {  
 sum += vector[j] \* matrix[j][i];  
 }  
 res[i] = sum;  
 }  
  
 ***return*** res;  
 }  
  
 ***private static int***[][] multiplyMatrices(***int***[][] firstMatrix, ***int***[][] secondMatrix) {  
 ***int*** y = firstMatrix.length;  
 ***int*** x = secondMatrix.length;  
 ***int*** z = secondMatrix[0].length;  
  
 ***int***[][] res = ***new int***[y][z];  
  
 ***for*** (***int*** i = 0; i < y; i++) {  
 ***for*** (***int*** j = 0; j < z; j++) {  
 res[i][j] = 0;  
 ***for*** (***int*** t = 0; t < x; t++) {  
 res[i][j] += firstMatrix[i][t] \* secondMatrix[t][j];  
 }  
 }  
 }  
 ***return*** res;  
 }  
  
 ***private static int***[] multScalarByVector(***int*** scalar, ***int***[] vector) {  
 ***int***[] res = ***new int***[vector.length];  
 ***for*** (***int*** i = 0; i < vector.length; i++) {  
 res[i] = scalar \* vector[i];  
 }  
 ***return*** res;  
 }  
 ***public static int***[] addVectors(***int***[] firstVector, ***int***[] secondVector) {  
 ***int***[] result = ***new int***[firstVector.length];  
 ***for*** (***int*** i = 0; i < firstVector.length; i++) {  
 result[i] = firstVector[i] + secondVector[i];  
 }  
 ***return*** result;  
 }  
  
  
 ***public static int***[] calculationThree(***int*** a, ***int*** e, ***int***[][] MV\_h, ***int***[][] MC\_h) {  
 ***return*** addVectors(multScalarByVector(a, multVectorByMatrix(B, MV\_h)), multScalarByVector(e, multVectorByMatrix(X, multiplyMatrices(MM, MC\_h))));  
 }  
  
 ***public static void*** combineA(***int***[] subVector, ***int*** start, ***int*** end) {  
 ***for*** (***int*** i = start; i < end; i++) {  
 A[i] = subVector[i - start];  
 }  
 }  
  
  
}

**SynchronizationMonitor.java**

***public class*** SynchronizationMonitor {  
 ***private int*** inputF = 0;  
 ***private int*** define\_a\_F = 0;  
 ***private int*** finalF = 0;  
  
 ***public synchronized void*** signalInput() {  
 inputF += 1;  
 ***if***(inputF == 4) notifyAll();  
 }  
 ***public synchronized void*** waitInput() {  
 ***try*** {  
 ***if***(inputF != 4) wait();  
 } ***catch*** (InterruptedException e) {  
 ***throw new*** RuntimeException(e);  
 }  
 }  
  
 ***public synchronized void*** signalDefineMin\_a() {  
 define\_a\_F += 1;  
 ***if***(define\_a\_F == 4) notifyAll();  
 }  
  
 ***public synchronized void*** waitDefineMin\_a() {  
 ***try*** {  
 ***if***(define\_a\_F != 4) wait();  
 } ***catch*** (InterruptedException e) {  
 ***throw new*** RuntimeException(e);  
 }  
 }  
  
 ***public synchronized void*** signalFinalCalc() {  
 finalF += 1;  
 ***if***(finalF == 3) notify();  
 }  
  
 ***public synchronized void*** waitFinalCalc() {  
 ***try*** {  
 ***if***(finalF != 3) wait();  
 } ***catch*** (InterruptedException e) {  
 ***throw new*** RuntimeException(e);  
 }  
 }  
}

**ResourceMonitor.java**

***public class*** ResourceMonitor {  
 ***private int*** e;  
 ***private int*** a = Integer.MAX\_VALUE;  
 ***public synchronized void*** write\_e(***int*** e) {  
 ***this***.e = e;  
 }  
 ***public synchronized int*** copy\_e() {  
 ***return*** e;  
 }  
 ***public synchronized void*** min\_a(***int*** ai) {  
 ***this***.a = Math.min(***this***.a, ai);  
 }  
 ***public synchronized int*** copy\_a() {  
 ***return*** a;  
 }  
}

**T1.java**

***import*** java.util.Arrays;  
  
***public class*** T1 ***extends*** Thread {  
 SynchronizationMonitor syncMon;  
 ResourceMonitor resMon;  
  
 ***public*** T1(SynchronizationMonitor syncMon, ResourceMonitor resMon) {  
 ***this***.syncMon = syncMon;  
 ***this***.resMon = resMon;  
 }  
  
 ***private static int*** start = 0;  
 ***private static int*** end = Data.H;  
  
 ***@Override  
 public void*** run() {  
 *// Повідомлення про початок виконання потоку T1* System.out.println("T1 started");  
  
 *// Введення MV, MC* Data.MV = Data.writeMatrix(Data.N);  
 Data.MC = Data.writeMatrix(Data.N);  
  
 *// Сигнал потокам T2, T3, T4 про завершення введення даних* syncMon.signalInput();  
 *// Очікування завершення введення даних потоками T2, T3, T4* syncMon.waitInput();  
  
 *// Обчислення 1* ***int***[] Z\_h = ***new int***[Data.H];  
 System.arraycopy(Data.B, start, Z\_h, 0, Data.H);  
  
 ***int*** a1 = Data.findMinVector(Z\_h);  
  
 *// Обчислення 2* resMon.min\_a(a1); *// КД1  
  
 // Сигнал потокам T2, T3, T4 про завершення обчислення 2* syncMon.signalDefineMin\_a();  
 *// Очікування завершення обчислення 2 у потоках T2, T3, T4* syncMon.waitDefineMin\_a();  
  
 *// Копіювання a1 = a* a1 = resMon.copy\_a(); *// КД2  
 // Копіювання e1 = e* ***int*** e1 = resMon.copy\_e(); *// КД3  
  
 // Обчислення 3* ***int***[][] MV\_h = Data.submatrixFromRangeOfColumns(Data.MV, start, end);  
 ***int***[][] MC\_h = Data.submatrixFromRangeOfColumns(Data.MC, start, end);  
  
 ***int***[] A\_h = Data.calculationThree(a1, e1, MV\_h, MC\_h);  
 Data.combineA(A\_h, start, end);  
  
 *// Сигнал потоку T3 про завершення обчислення 3* syncMon.signalFinalCalc();  
 *// Повідомлення про завершення виконання потоку T1* System.out.println("T1 finished");  
 }  
}

**T2.java**

***public class*** T2 ***extends*** Thread {  
 SynchronizationMonitor syncMon;  
 ResourceMonitor resMon;  
 ***public*** T2( SynchronizationMonitor syncMon, ResourceMonitor resMon ) {  
 ***this***.syncMon = syncMon;  
 ***this***.resMon = resMon;  
 }  
  
 ***private static int*** start = Data.H;  
 ***private static int*** end = Data.H \* 2;  
 ***@Override  
 public void*** run() {  
 *// Повідомлення про початок виконання потоку T2* System.out.println("T2 started");  
  
 *// Введення MM, e* Data.MM = Data.writeMatrix(Data.N);  
 Data.e = Data.writeScalar();  
 resMon.write\_e(Data.e);  
  
 *// Сигнал потокам T1, T3, T4 про завершення введення даних* syncMon.signalInput();  
 *// Очікування завершення введення даних потоками T1, T3, T4* syncMon.waitInput();  
  
 *// Обчислення 1* ***int***[] Z\_h = ***new int***[Data.H];  
 System.arraycopy(Data.B, start, Z\_h, 0, Data.H);  
  
 ***int*** a2 = Data.findMinVector(Z\_h);  
  
 *// Обчислення 2* resMon.min\_a(a2); *// КД1  
  
 // Сигнал потокам T1, T3, T4 про завершення обчислення 2* syncMon.signalDefineMin\_a();  
 *// Очікування завершення обчислення 2 у потоках T1, T3, T4* syncMon.waitDefineMin\_a();  
  
 *// Копіювання a2 = a* a2 = resMon.copy\_a(); *// КД2  
 // Копіювання e2 = e* ***int*** e2 = resMon.copy\_e(); *// КД3  
  
 // Обчислення 3* ***int***[][] MV\_h = Data.submatrixFromRangeOfColumns(Data.MV, start, end);  
 ***int***[][] MC\_h = Data.submatrixFromRangeOfColumns(Data.MC, start, end);  
  
 ***int***[] A\_h = Data.calculationThree(a2, e2, MV\_h, MC\_h);  
 Data.combineA(A\_h, start, end);  
  
 *// Сигнал потоку T3 про завершення обчислення 3* syncMon.signalFinalCalc();  
 *// Повідомлення про завершення виконання потоку T2* System.out.println("T2 finished");  
 }  
}

**T3.java**

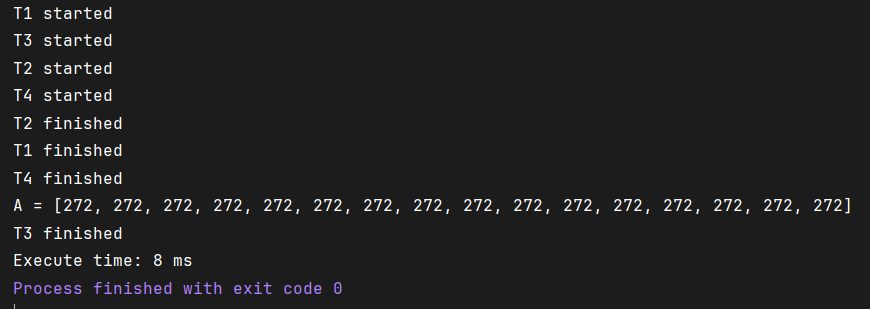
***import*** java.util.Arrays;  
  
***public class*** T3 ***extends*** Thread {  
 SynchronizationMonitor syncMon;  
 ResourceMonitor resMon;  
 ***public*** T3( SynchronizationMonitor syncMon, ResourceMonitor resMon ) {  
 ***this***.syncMon = syncMon;  
 ***this***.resMon = resMon;  
 }  
 ***private static int*** start = Data.H \* 2;  
 ***private static int*** end = Data.H \* 3;  
 ***@Override  
 public void*** run() {  
 *// Повідомлення про початок виконання потоку T3* System.out.println("T3 started");  
  
 *// Введення A, Z* Data.A = Data.writeVector(Data.N);  
 Data.Z = Data.writeVector(Data.N);  
  
 *// Сигнал потокам T1, T2, T4 про завершення введення даних* syncMon.signalInput();  
 *// Очікування завершення введення даних потоками T1, T2, T4* syncMon.waitInput();  
  
 *// Обчислення 1* ***int***[] Z\_h = ***new int***[Data.H];  
 System.arraycopy(Data.B, start, Z\_h, 0, Data.H);  
  
 ***int*** a3 = Data.findMinVector(Z\_h);  
  
 *// Обчислення 2* resMon.min\_a(a3); *// КД1  
  
 // Сигнал потокам T1, T2, T4 про завершення обчислення 2* syncMon.signalDefineMin\_a();  
 *// Очікування завершення обчислення 2 у потоках T1, T2, T4* syncMon.waitDefineMin\_a();  
  
 *// Копіювання a3 = a* a3 = resMon.copy\_a(); *// КД2  
 // Копіювання e3 = e* ***int*** e3 = resMon.copy\_e(); *// КД3  
  
 // Обчислення 3* ***int***[][] MV\_h = Data.submatrixFromRangeOfColumns(Data.MV, start, end);  
 ***int***[][] MC\_h = Data.submatrixFromRangeOfColumns(Data.MC, start, end);  
  
 ***int***[] A\_h = Data.calculationThree(a3, e3, MV\_h, MC\_h);  
 Data.combineA(A\_h, start, end);  
  
 *// Очікування завершення обчислення 3 у потоках T1, T2, T4* syncMon.waitFinalCalc();  
  
 *// Виведення результату A* System.out.printf("A = %s \n", Arrays.toString(Data.A));  
 *// Повідомлення про завершення виконання потоку T3* System.out.println("T3 finished");  
 }  
}

**T4.java**

***public class*** T4 ***extends*** Thread {  
 SynchronizationMonitor syncMon;  
 ResourceMonitor resMon;  
 ***public*** T4( SynchronizationMonitor syncMon, ResourceMonitor resMon ) {  
 ***this***.syncMon = syncMon;  
 ***this***.resMon = resMon;  
 }  
 ***private static int*** start = Data.H \* 3;  
 ***private static int*** end = Data.N;  
 ***@Override  
 public void*** run() {  
 *// Повідомлення про початок виконання потоку T4* System.out.println("T4 started");  
  
 *// Введення B, X* Data.B = Data.writeVector(Data.N);  
 Data.X = Data.writeVector(Data.N);  
  
 *// Сигнал потокам T1, T2, T3 про завершення введення даних* syncMon.signalInput();  
 *// Очікування завершення введення даних потоками T1, T2, T3* syncMon.waitInput();  
  
 *// Обчислення 1* ***int***[] Z\_h = ***new int***[Data.H];  
 System.arraycopy(Data.B, start, Z\_h, 0, Data.H);  
  
 ***int*** a4 = Data.findMinVector(Z\_h);  
  
 *// Обчислення 2* resMon.min\_a(a4); *// КД1  
  
 // Сигнал потокам T1, T2, T3 про завершення обчислення 2* syncMon.signalDefineMin\_a();  
 syncMon.waitDefineMin\_a();  
  
 *// Копіювання a4 = a* a4 = resMon.copy\_a(); *// КД2  
 // Копіювання e4 = e* ***int*** e4 = resMon.copy\_e(); *// КД3  
  
 // Обчислення 3* ***int***[][] MV\_h = Data.submatrixFromRangeOfColumns(Data.MV, start, end);  
 ***int***[][] MC\_h = Data.submatrixFromRangeOfColumns(Data.MC, start, end);  
  
 ***int***[] A\_h = Data.calculationThree(a4, e4, MV\_h, MC\_h);  
 Data.combineA(A\_h, start, end);  
  
 *// Сигнал потоку T3 про завершення обчислення 3* syncMon.signalFinalCalc();  
 *// Повідомлення про завершення виконання потоку T4* System.out.println("T4 finished");  
 }  
}

**Результат виконання:**

**При n = 16**



**При n = 1500:**

**Для 4 ядер - 2274ms**

**Для 1 ядра - 4804ms**

**Таким чином, коефіцієнт прискорення Кп = 4.804 с / 2.274 с = 2,113.**