

Міністерство освіти і науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота № 3

з курсу “ Паралельне програмування-2. Програмування для паралельних систем”

на тему:

“ Java. Монітори ”

Виконав: студент ІІI курсу групи ІП – 42

Водотієць Денис Ігорович

Київ – 2017

**Технічне завдання**

Структура ПКС:



Математична задача: MA = (B \* C) \* MO + max(Z) \* (MS \* MR).

Мова та бібліотека програмування: Java.

**Виконання**

Етап 1. Розробка паралельного математичного алгоритму

1. a*i* = B*H* \* C*H*, i = 1..P;

a = a + a*i*;

2. b*i* = max(Z*H*), i = 1..P;

b = mix(b, b*i*);

3. MA*H* = a \* MO*H* + b \* (MS*H* \* MR).

Де H = N / P, де P – кількість процесів,

B*H* - H елементів вектора B,

С*H* - H елементів вектора С,

Z*H* - H елементів вектора Z,

MA*H* - H рядків матриці MА,

MS*H* - H рядків матриці МS.

ЗР: a, b, MR.

Етап 2. Розробка алгоритму потоків

Потік Т1:

|  |  |
| --- | --- |
| 1. Введення MR, C, Z; |  |
| 2. Сигнал Т2-6 про завершення введення; | S*2-6:1* |
| 3. Очікувати завершення введення в T2, Т6; | W*2,6:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*2-6:2* |
| 7. Рахунок 3: b*i* = max(Z*H*); |  |
| 8. Рахунок 4: b = max(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*2-6:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=2..6) | W*2-6:2* |
| 11. Копіювати а1 = а, b1 = b, MR1 = MR; | КУ |
| 12. Рахунок 5: MA*H* = a1 \* MO*H* + b1 \* (MS*H* \* MR1); |  |
| 13. Cигнал про завершення рахунку 5 | S2-6 |

Потік Т2:

|  |  |
| --- | --- |
| 1. Введення B, MS; |  |
| 2. Сигнал Т1, Т3-6 про завершення введення; | S*1,3-6:1* |
| 3. Очікувати завершення введення в T1, Т6; | W*1,6:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*1,3-6:2* |
| 7. Рахунок 3: b*i* = max(Z*H*); |  |
| 8. Рахунок 4: b = max(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*1,3-6:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=1..6, i ≠ 2) | W*1,3-6:2* |
| 11. Копіювати а2 = а, b2 = b, MR2 = MR; | КУ |
| 12. Рахунок 5: MA*H* = a2 \* MO*H* + b2 \* (MS*H* \* MR2); |  |
| 13. Сигнал про завершення рахунку 5 | S*1,3-6* |

Потік Т3:

|  |  |
| --- | --- |
| 1. Очікувати завершення введення в T1, Т2, Т6; | W*1,2,6:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*1,2,4-6:2* |
| 7. Рахунок 3: b*i* = max(Z*H*); |  |
| 8. Рахунок 4: b = max(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*1,2,4-6:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=1..6, i ≠ 3) | W*1,2,4-6:2* |
| 11. Копіювати а3 = а, b3 = b, MR3 = MR; | КУ |
| 12. Рахунок 5: MA*H* = a3 \* MO*H* + b3 \* (MS*H* \* MR3); |  |
| 13. Сигнал про завершення рахунку 5 | S*1,2,4-6:4* |

Потік Т4:

|  |  |
| --- | --- |
| 1. Очікувати завершення введення в T1, Т2, Т6; | W*1,2,6:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*1-3,5,6:2* |
| 7. Рахунок 3: b*i* = max(Z*H*); |  |
| 8. Рахунок 4: b = max(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*1-3,5,6:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=1..6, i ≠ 4) | W*1-3,5,6:2* |
| 11. Копіювати а4 = а, b4 = b, MR4 = MR; | КУ |
| 12. Рахунок 5: MA*H* = a4 \* MO*H* + b4 \* (MS*H* \* MR4); |  |
| 13. Сигнал про завершення рахунку 5 | S*1-3,5,6:4* |

Потік Т5:

|  |  |
| --- | --- |
| 1. Очікувати завершення введення в T1, Т2, Т6; | W*1,2,6:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*1-4,6:2* |
| 7. Рахунок 3: b*i* = max(Z*H*); |  |
| 8. Рахунок 4: b = max(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*1-4,6:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=1..6, i ≠ 5) | W*1-3,5,6:2* |
| 11. Копіювати а5 = а, b5 = b, MR5 = MR; | КУ |
| 12. Рахунок 5: MA*H* = a5 \* MO*H* + b5 \* (MS*H* \* MR5); |  |
| 13. Сигнал про завершення рахунку 5 | S*1-4,6:4* |

Потік Т6:

|  |  |
| --- | --- |
| 1. Введення MO; |  |
| 2. Сигнал Т1-5 про завершення введення; | S*1-5:1* |
| 3. Очікувати завершення введення в T1, Т2; | W*1,2:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*1-5:2* |
| 7. Рахунок 3: b*i* = max(Z*H*); |  |
| 8. Рахунок 4: b = maxb, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*1-5:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=1..5) | W*1-5:2* |
| 11. Копіювати а6 = а, b6 = b, MR6 = MR; | КУ |
| 12. Рахунок 5: MA*H* = a6 \* MO*H* + b4 \* (MS*H* \* MR6); |  |
| 13. Очікувати завершення рахунку 5 від Ti (i=1..5) | W*1-6:3* |
| 14. Вивести MA |  |

Етап 3. Розробка схеми взаємодії потоків



Етап 4. Розробка програми

**Director.java**

package com.vodotiiets.directors;

import com.vodotiiets.Tasks.\*;

import com.vodotiiets.controllers.DataController;

import com.vodotiiets.controllers.TaskController;

import com.vodotiiets.primitives.Matrix;

import com.vodotiiets.primitives.Vector;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class Director {

public static final int N = 10;

public static final int P = 6;

public static final int H = N / P;

public static int inputCount = 3;

public static Vector B = new Vector(N);

public static Vector C = new Vector(N);

public static Vector Z = new Vector(N);

public static Matrix MO = new Matrix(N);

public static Matrix MS = new Matrix(N);

public static Matrix MR = new Matrix(N);

public static Matrix MA = new Matrix(N);

public static void main(String[] args) {

Thread[] tasks = new Thread[P];

DataController dm = new DataController();

TaskController tm = new TaskController();

tasks[0] = new Thread(new Task1(dm, tm));

tasks[1] = new Thread(new Task2(dm, tm));

tasks[2] = new Thread(new Task3(dm, tm));

tasks[3] = new Thread(new Task4(dm, tm));

tasks[4] = new Thread(new Task5(dm, tm));

tasks[5] = new Thread(new Task6(dm, tm));

for (int i = 0; i < P; i++) {

tasks[i].start();

}

}

}

**DataController.java**

package com.vodotiiets.controllers;

import com.vodotiiets.directors.Director;

import com.vodotiiets.primitives.Matrix;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class DataController {

private int a;

private int b;

private Matrix MR;

public DataController() {

a = 0;

b = Integer.MIN\_VALUE;

MR = new Matrix(Director.N);

}

public synchronized void calculateA(int a) {

this.a += a;

}

public int copyA() {

return a;

}

public synchronized void calculateB(int b) {

this.b = b > this.b ? b : this.b;

}

public int copyB() {

return b;

}

public synchronized void inputMR() {

MR.fill();

}

public Matrix copyMR() {

return new Matrix(MR);

}

}

**TaskController.java**

package com.vodotiiets.controllers;

import com.vodotiiets.directors.Director;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class TaskController {

private int inputCount;

private int aCount;

private int bCount;

private int calculateCount;

public synchronized void signalInputDone() {

if(++inputCount == Director.inputCount) {

notifyAll();

}

}

public synchronized void waitInput() {

while (inputCount < Director.inputCount) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public synchronized void signalCalculateADone() {

if(++aCount == Director.P) {

notifyAll();

}

}

public synchronized void waitCalculateA() {

while (aCount < Director.P){

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public synchronized void signalCalculateBDone() {

if (++bCount == Director.P) {

notifyAll();

}

}

public synchronized void waitCalculateB() {

while (bCount < Director.P) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public synchronized void signalCalculateDone() {

if (++calculateCount == Director.P - 1) {

notifyAll();

}

}

public synchronized void waitCalculate() {

while (calculateCount < Director.P - 1) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**Task1.java**

package com.vodotiiets.Tasks;

import com.vodotiiets.Utils.Util;

import com.vodotiiets.controllers.DataController;

import com.vodotiiets.controllers.TaskController;

import com.vodotiiets.directors.Director;

import com.vodotiiets.primitives.Matrix;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class Task1 implements Runnable {

private int startIndex;

private int endIndex;

private DataController dataController;

private TaskController taskController;

public Task1(DataController dataController, TaskController taskController) {

this.dataController = dataController;

this.taskController = taskController;

startIndex = 0;

endIndex = Director.H;

}

@Override

public void run() {

//input

dataController.inputMR();

Director.C.fill();

Director.Z.fill();

taskController.signalInputDone();

taskController.waitInput();

//Calculate alpha

int a\_i = 0;

for (int i = startIndex; i < endIndex; i++) {

a\_i += Director.B.get(i) \* Director.C.get(i);

}

dataController.calculateA(a\_i);

taskController.signalCalculateADone();

//Calculate beta

int b\_i = Integer.MIN\_VALUE;

for(int i = startIndex; i < endIndex; i++) {

b\_i = b\_i < Director.Z.get(i) ? Director.Z.get(i) : b\_i;

}

dataController.calculateB(b\_i);

taskController.signalCalculateBDone();

taskController.waitCalculateA();

taskController.waitCalculateB();

//Copy data

int a = dataController.copyA();

int b = dataController.copyB();

Matrix MR = dataController.copyMR();

//Calculate

Util.calculate(startIndex, endIndex, a, b, MR);

taskController.signalCalculateDone();

}

}

**Task2.java**

package com.vodotiiets.Tasks;

import com.vodotiiets.Utils.Util;

import com.vodotiiets.controllers.DataController;

import com.vodotiiets.controllers.TaskController;

import com.vodotiiets.directors.Director;

import com.vodotiiets.primitives.Matrix;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class Task2 implements Runnable {

private int startIndex;

private int endIndex;

private DataController dataController;

private TaskController taskController;

public Task2(DataController dataController, TaskController taskController) {

this.dataController = dataController;

this.taskController = taskController;

startIndex = Director.H;

endIndex = Director.H \* 2;

}

@Override

public void run() {

//input

dataController.inputMR();

Director.B.fill();

Director.MS.fill();

taskController.signalInputDone();

taskController.waitInput();

//Calculate alpha

int a\_i = 0;

for (int i = startIndex; i < endIndex; i++) {

a\_i += Director.B.get(i) \* Director.C.get(i);

}

dataController.calculateA(a\_i);

taskController.signalCalculateADone();

//Calculate beta

int b\_i = Integer.MIN\_VALUE;

for (int i = startIndex; i < endIndex; i++) {

b\_i = b\_i < Director.Z.get(i) ? Director.Z.get(i) : b\_i;

}

dataController.calculateB(b\_i);

taskController.signalCalculateBDone();

taskController.waitCalculateA();

taskController.waitCalculateB();

//Copy data

int a = dataController.copyA();

int b = dataController.copyB();

Matrix MR = dataController.copyMR();

//Calculate

Util.calculate(startIndex, endIndex, a, b, MR);

taskController.signalCalculateDone();

}

}

**Task3.java**

package com.vodotiiets.Tasks;

import com.vodotiiets.Utils.Util;

import com.vodotiiets.controllers.DataController;

import com.vodotiiets.controllers.TaskController;

import com.vodotiiets.directors.Director;

import com.vodotiiets.primitives.Matrix;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class Task3 implements Runnable {

private int startIndex;

private int endIndex;

private DataController dataController;

private TaskController taskController;

public Task3(DataController dataController, TaskController taskController) {

this.dataController = dataController;

this.taskController = taskController;

startIndex = Director.H \* 2;

endIndex = Director.H \* 3;

}

@Override

public void run() {

Util.run(taskController, dataController, startIndex, endIndex);

taskController.signalCalculateDone();

}

}

**Task4.java**

package com.vodotiiets.Tasks;

import com.vodotiiets.Utils.Util;

import com.vodotiiets.controllers.DataController;

import com.vodotiiets.controllers.TaskController;

import com.vodotiiets.directors.Director;

import com.vodotiiets.primitives.Matrix;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class Task4 implements Runnable {

private int startIndex;

private int endIndex;

private DataController dataController;

private TaskController taskController;

public Task4(DataController dataController, TaskController taskController) {

this.dataController = dataController;

this.taskController = taskController;

startIndex = Director.H \* 3;

endIndex = Director.H \* 4;

}

@Override

public void run() {

Util.run(taskController, dataController, startIndex, endIndex);

taskController.signalCalculateDone();

}

}

**Task5.java**

package com.vodotiiets.Tasks;

import com.vodotiiets.Utils.Util;

import com.vodotiiets.controllers.DataController;

import com.vodotiiets.controllers.TaskController;

import com.vodotiiets.directors.Director;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class Task5 implements Runnable{

private int startIndex;

private int endIndex;

private DataController dataController;

private TaskController taskController;

public Task5(DataController dataController, TaskController taskController) {

this.dataController = dataController;

this.taskController = taskController;

startIndex = Director.H \* 4;

endIndex = Director.H \* 5;

}

@Override

public void run() {

Util.run(taskController, dataController, startIndex, endIndex);

taskController.signalCalculateDone();

}

}

**Task6.java**

package com.vodotiiets.Tasks;

import com.vodotiiets.Utils.Util;

import com.vodotiiets.controllers.DataController;

import com.vodotiiets.controllers.TaskController;

import com.vodotiiets.directors.Director;

import com.vodotiiets.primitives.Matrix;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class Task6 implements Runnable {

private int startIndex;

private int endIndex;

private DataController dataController;

private TaskController taskController;

public Task6(DataController dataController, TaskController taskController) {

this.dataController = dataController;

this.taskController = taskController;

startIndex = Director.H \* 5;

endIndex = Director.N;

}

@Override

public void run() {

//input

Director.MO.fill();

taskController.signalInputDone();

taskController.waitInput();

//Calculate alpha

int a\_i = 0;

for (int i = startIndex; i < endIndex; i++) {

a\_i += Director.B.get(i) \* Director.C.get(i);

}

dataController.calculateA(a\_i);

taskController.signalCalculateADone();

//Calculate beta

int b\_i = Integer.MIN\_VALUE;

for(int i = startIndex; i < endIndex; i++) {

b\_i = b\_i < Director.Z.get(i) ? Director.Z.get(i) : b\_i;

}

dataController.calculateB(b\_i);

taskController.signalCalculateBDone();

taskController.waitCalculateA();

taskController.waitCalculateB();

//Copy data

int a = dataController.copyA();

int b = dataController.copyB();

Matrix MR = dataController.copyMR();

//Calculate

Util.calculate(startIndex, endIndex, a, b, MR);

taskController.waitCalculate();

System.out.print("Result ");

if (Director.N > 10) {

System.out.println("is to large!");

} else {

System.out.println("\n" + Director.MA);

}

}

}

**Util.java**

package com.vodotiiets.Utils;

import com.vodotiiets.controllers.DataController;

import com.vodotiiets.controllers.TaskController;

import com.vodotiiets.directors.Director;

import com.vodotiiets.primitives.Matrix;

import com.vodotiiets.primitives.Vector;

/\*\*

\* Created by Denys Vodotiiets.

\*/

public class Util {

public static void calculate(int startIndex, int endIndex, int a, int b, Matrix MR) {

for (int i = startIndex; i < endIndex; i++) {

//MS\*MR

Vector temp = new Vector(Director.N);

for (int j = 0; j < Director.N; j++) {

int element = 0;

for(int k = 0; k < Director.N; k++) {

element += Director.MS.get(i, k) \* MR.get(k, j);

}

temp.set(j, element);

}

for (int j = 0; j < Director.N; j++) {

//b\*(MS\*MR)

int sum = temp.get(i) \* b;

//a\*MO\*b(MS\*MR)

sum += a \* Director.MO.get(i, j);

Director.MA.set(i, j, sum);

}

}

}

public static void run(TaskController taskController, DataController dataController, int startIndex, int endIndex) {

taskController.waitInput();

//Calculate alpha

int a\_i = 0;

for (int i = startIndex; i < endIndex; i++) {

a\_i += Director.B.get(i) \* Director.C.get(i);

}

dataController.calculateA(a\_i);

taskController.signalCalculateADone();

//Calculate beta

int b\_i = Integer.MIN\_VALUE;

for (int i = startIndex; i < endIndex; i++) {

b\_i = b\_i < Director.Z.get(i) ? Director.Z.get(i) : b\_i;

}

dataController.calculateB(b\_i);

taskController.signalCalculateBDone();

taskController.waitCalculateA();

taskController.waitCalculateB();

//Copy data

int a = dataController.copyA();

int b = dataController.copyB();

Matrix MR = dataController.copyMR();

//Calculate

Util.calculate(startIndex, endIndex, a, b, MR);

}

}