

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

Кафтанатій Богдан Сергійович

Київ – 2017

**Технічне завдання**

Структура ПКС:



Математична задача: A = (B \* C) \* E + min(Z) \* S \* (MO \* MK).

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* = min(Z*H*), i = 1..P;

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

3. A*H* = a \* E*H* + b \* S \* (MO*H* \* MK).

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

A*H* - H елементів вектора А,

MO*H* - H рядків матриці МO.

ОР: a, b, S, MK.

Етап 2. Розробка алгоритму потоків

Потік Т1:

|  |  |
| --- | --- |
| 1. Введення MK, B; |  |
| 2. Сигнал Т2, Т3, Т4 про завершення введення; | S*2,3,4:1* |
| 3. Очікувати завершення введення в T2, Т4; | W*2,4:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*2,3,4:2* |
| 7. Рахунок 3: b*i* = min(Z*H*); |  |
| 8. Рахунок 4: b = min(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*2,3,4:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=2..4) | W*2,3,4:2* |
| 11. Копіювати а1 = а, b1 = b, S1 = S, MK1 = MK; | КУ |
| 12. Рахунок 5: A*H* = a1 \* E*H* + b1 \* S1 \* (MO*H* \* MK1); |  |
| 13. Очікувати завершення рахунку 5 від Ti (i=2..4) | W*2,3,4:3* |
| 14. Вивести А. |  |

Потік Т2:

|  |  |
| --- | --- |
| 1. Введення MO, Z; |  |
| 2. Сигнал Т1, Т3, Т4 про завершення введення; | S*1,3,4:1* |
| 3. Очікувати завершення введення в T1, Т4; | W*1,4:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*1,3,4:2* |
| 7. Рахунок 3: b*i* = min(Z*H*); |  |
| 8. Рахунок 4: b = min(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*1,3,4:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=1..4, i ≠2) | W*1,3,4:2* |
| 11. Копіювати а2 = а, b2 = b, S2 = S, MK2 = MK; | КУ |
| 12. Рахунок 5: A*H* = a2 \* E*H* + b2 \* S2 \* (MO*H* \* MK2); |  |
| 13. Сигнал про завершення рахунку 5 | S*1:4* |

Потік Т3:

|  |  |
| --- | --- |
| 1. Очікувати завершення введення в T1, Т2, Т4; | W*1,2,4:1* |
| 4. Рахунок 1: a*i* = B*H* \* C*H;* |  |
| 5. Рахунок 2: a = a + a*i;* | КУ |
| 6. Сигнал про завершення рахунку 2; | S*1,2,4:2* |
| 7. Рахунок 3: b*i* = min(Z*H*); |  |
| 8. Рахунок 4: b = min(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*1,2,4:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=1..4, i ≠3) | W*1,2,4:2* |
| 11. Копіювати а3 = а, b3 = b, S3 = S, MK3 = MK; | КУ |
| 12. Рахунок 5: A*H* = a3 \* E*H* + b3 \* S3 \* (MO*H* \* MK3); |  |
| 13. Сигнал про завершення рахунку 5 | S*1:4* |

Потік Т4:

|  |  |
| --- | --- |
| 1. Введення С, E, S; |  |
| 2. Сигнал Т1, Т2, Т3 про завершення введення; | S*1,2,3:1* |
| 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*1,2,3:2* |
| 7. Рахунок 3: b*i* = min(Z*H*); |  |
| 8. Рахунок 4: b = min(b, b*i*); | КУ |
| 9. Сигнал про завершення рахунку 4; | S*1,2,3:3* |
| 10. Очікувати завершення рахунків 2 і 4 від Ti (i=1..4, i ≠2) | W*1,2,3:2* |
| 11. Копіювати а4 = а, b4 = b, S4 = S, MK4 = MK; | КУ |
| 12. Рахунок 5: A*H* = a4 \* E*H* + b4 \* S4 \* (MO*H* \* MK4); |  |
| 13. Сигнал про завершення рахунку 5 | S*1:4* |

Етап 3. Розробка схеми взаємодії потоків



Етап 4. Розробка програми

**Main.java**

package com.kaftanatiy.Lab3Java;

import com.kaftanatiy.Lab3Java.primitives.\*;

import com.kaftanatiy.Lab3Java.threads.\*;

/\*\*

\* Lab 3. Java

\* A = (B \* C) \* E + min(Z) \* S \* (MO \* MK)

\* @author Bogdan Kaftanatiy IP-42

\*/

public class Main {

public static int N = 4;

public static final int P = 4;

public static int H = N / P;

public static MyVector A = new MyVector(N);

public static MyVector B = new MyVector(N);

public static MyVector C = new MyVector(N);

public static MyVector E = new MyVector(N);

public static MyVector Z = new MyVector(N);

public static MyMatrix MO = new MyMatrix(N);

public static void main(String[] args) {

Thread[] tasks = new Thread[P];

DataControl dm = new DataControl();

TaskControl tm = new TaskControl();

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));

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

tasks[i].start();

}

}

}

**DataControl.java**

package com.kaftanatiy.Lab3Java.threads;

import com.kaftanatiy.Lab3Java.Main;

import com.kaftanatiy.Lab3Java.primitives.MyMatrix;

import com.kaftanatiy.Lab3Java.primitives.MyVector;

/\*\*

\* Monitor class to synchronize access to data

\* @author Bogdan Kaftanatiy

\*/

public class DataControl {

private int alpha;

private int beta;

private MyVector S;

private MyMatrix MK;

public DataControl() {

alpha = 0;

beta = Integer.MAX\_VALUE;

S = new MyVector(Main.N);

MK = new MyMatrix(Main.N);

}

public synchronized void calculateAlpha(int a) {

alpha += a;

}

public int copyAlpha() {

return alpha;

}

public synchronized void calculateBeta(int b) {

if(b < beta)

beta = b;

}

public int copyBeta() {

return beta;

}

public synchronized void inputS() {

S.fill();

}

public MyVector copyS() {

return new MyVector(S);

}

public synchronized void inputMK() {

MK.fill();

}

public MyMatrix copyMK() {

return new MyMatrix(MK);

}

}

**TaskControl.java**

package com.kaftanatiy.Lab3Java.threads;

/\*\*

\* Monitor class to synchronize threads

\* @author Bogdan Kaftanatiy

\*/

public class TaskControl {

private int inputCount = 0;

private int alphaCount = 0;

private int betaCount = 0;

private int calcCount = 0;

public synchronized void signalInputDone() {

inputCount++;

if(inputCount == 3)

notifyAll();

}

public synchronized void waitInput() {

while (inputCount < 3) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public synchronized void signalCalcAlphaDone() {

alphaCount++;

if(alphaCount == 4)

notifyAll();

}

public synchronized void waitCalcAlpha() {

while (alphaCount < 4){

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public synchronized void signalCalcBetaDone() {

betaCount++;

if (betaCount == 4)

notifyAll();

}

public synchronized void waitCalcBeta() {

while (betaCount < 4) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public synchronized void signalCalcDone() {

calcCount++;

if (calcCount == 3)

notifyAll();

}

public synchronized void waitCalc() {

while (calcCount < 3) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**Task1.java**

package com.kaftanatiy.Lab3Java.threads;

import com.kaftanatiy.Lab3Java.Main;

import com.kaftanatiy.Lab3Java.primitives.MyMatrix;

import com.kaftanatiy.Lab3Java.primitives.MyVector;

public class Task1 implements Runnable {

private int startIndex;

private int endIndex;

private DataControl dataMonitor;

private TaskControl taskMonitor;

private int alpha1;

private int beta1;

private MyVector S1;

private MyMatrix MK1;

public Task1(DataControl dm, TaskControl tm) {

dataMonitor = dm;

taskMonitor = tm;

startIndex = 0;

endIndex = Main.H;

}

@Override

public void run() {

//input

dataMonitor.inputMK();

Main.B.fill();

taskMonitor.signalInputDone();

taskMonitor.waitInput();

//Calculate alpha

int alpha\_i = 0;

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

alpha\_i += Main.B.get(i) \* Main.C.get(i);

}

dataMonitor.calculateAlpha(alpha\_i);

taskMonitor.signalCalcAlphaDone();

//Calculate beta

int beta\_i = Integer.MAX\_VALUE;

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

if(Main.Z.get(i) < beta\_i)

beta\_i = Main.Z.get(i);

}

dataMonitor.calculateBeta(beta\_i);

taskMonitor.signalCalcBetaDone();

taskMonitor.waitCalcAlpha();

taskMonitor.waitCalcBeta();

//Copy data

alpha1 = dataMonitor.copyAlpha();

beta1 = dataMonitor.copyBeta();

S1 = dataMonitor.copyS();

MK1 = dataMonitor.copyMK();

//Calculate

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

//MO\*MK

MyVector temp = new MyVector(Main.N);

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

int element = 0;

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

element += Main.MO.get(i, k) \* MK1.get(k, j);

}

temp.set(j, element);

}

//S\*(MO\*MK)

int sum = 0;

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

sum += S1.get(j) \* temp.get(j);

}

//b\*S\*(MO\*MK)

sum \*= beta1;

//a\*E+b\*S\*(MO\*MK)

sum = alpha1 \* Main.E.get(i) + sum;

Main.A.set(i, sum);

}

//End calculate

taskMonitor.waitCalc();

System.out.println("Result:");

if(Main.N > 10)

System.out.println(" is too large..");

else

System.out.println("A: " + Main.A);

}

}

**Task2.java**

package com.kaftanatiy.Lab3Java.threads;

import com.kaftanatiy.Lab3Java.Main;

import com.kaftanatiy.Lab3Java.primitives.MyMatrix;

import com.kaftanatiy.Lab3Java.primitives.MyVector;

public class Task2 implements Runnable {

private int startIndex;

private int endIndex;

private DataControl dataMonitor;

private TaskControl taskMonitor;

private int alpha2;

private int beta2;

private MyVector S2;

private MyMatrix MK2;

public Task2(DataControl dm, TaskControl tm) {

dataMonitor = dm;

taskMonitor = tm;

startIndex = Main.H;

endIndex = Main.H \* 2;

}

@Override

public void run() {

//input

Main.MO.fill();

Main.Z.fill();

taskMonitor.signalInputDone();

taskMonitor.waitInput();

//Calculate alpha

int alpha\_i = 0;

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

alpha\_i += Main.B.get(i) \* Main.C.get(i);

}

dataMonitor.calculateAlpha(alpha\_i);

taskMonitor.signalCalcAlphaDone();

//Calculate beta

int beta\_i = Integer.MAX\_VALUE;

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

if(Main.Z.get(i) < beta\_i)

beta\_i = Main.Z.get(i);

}

dataMonitor.calculateBeta(beta\_i);

taskMonitor.signalCalcBetaDone();

taskMonitor.waitCalcAlpha();

taskMonitor.waitCalcBeta();

//Copy data

alpha2 = dataMonitor.copyAlpha();

beta2 = dataMonitor.copyBeta();

S2 = dataMonitor.copyS();

MK2 = dataMonitor.copyMK();

//Calculate

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

//MO\*MK

MyVector temp = new MyVector(Main.N);

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

int element = 0;

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

element += Main.MO.get(i, k) \* MK2.get(k, j);

}

temp.set(j, element);

}

//S\*(MO\*MK)

int sum = 0;

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

sum += S2.get(j) \* temp.get(j);

}

//b\*S\*(MO\*MK)

sum \*= beta2;

//a\*E+b\*S\*(MO\*MK)

sum = alpha2 \* Main.E.get(i) + sum;

Main.A.set(i, sum);

}

//End calculate

taskMonitor.signalCalcDone();

}

}

**Task3.java**

package com.kaftanatiy.Lab3Java.threads;

import com.kaftanatiy.Lab3Java.Main;

import com.kaftanatiy.Lab3Java.primitives.MyMatrix;

import com.kaftanatiy.Lab3Java.primitives.MyVector;

public class Task3 implements Runnable {

private int startIndex;

private int endIndex;

private DataControl dataMonitor;

private TaskControl taskMonitor;

private int alpha3;

private int beta3;

private MyVector S3;

private MyMatrix MK3;

public Task3(DataControl dm, TaskControl tm) {

dataMonitor = dm;

taskMonitor = tm;

startIndex = Main.H \* 2;

endIndex = Main.H \* 3;

}

@Override

public void run() {

taskMonitor.waitInput();

//Calculate alpha

int alpha\_i = 0;

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

alpha\_i += Main.B.get(i) \* Main.C.get(i);

}

dataMonitor.calculateAlpha(alpha\_i);

taskMonitor.signalCalcAlphaDone();

//Calculate beta

int beta\_i = Integer.MAX\_VALUE;

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

if(Main.Z.get(i) < beta\_i)

beta\_i = Main.Z.get(i);

}

dataMonitor.calculateBeta(beta\_i);

taskMonitor.signalCalcBetaDone();

taskMonitor.waitCalcAlpha();

taskMonitor.waitCalcBeta();

//Copy data

alpha3 = dataMonitor.copyAlpha();

beta3 = dataMonitor.copyBeta();

S3 = dataMonitor.copyS();

MK3 = dataMonitor.copyMK();

//Calculate

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

//MO\*MK

MyVector temp = new MyVector(Main.N);

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

int element = 0;

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

element += Main.MO.get(i, k) \* MK3.get(k, j);

}

temp.set(j, element);

}

//S\*(MO\*MK)

int sum = 0;

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

sum += S3.get(j) \* temp.get(j);

}

//b\*S\*(MO\*MK)

sum \*= beta3;

//a\*E+b\*S\*(MO\*MK)

sum = alpha3 \* Main.E.get(i) + sum;

Main.A.set(i, sum);

}

//End calculate

taskMonitor.signalCalcDone();

}

}

**Task4.java**

package com.kaftanatiy.Lab3Java.threads;

import com.kaftanatiy.Lab3Java.Main;

import com.kaftanatiy.Lab3Java.primitives.MyMatrix;

import com.kaftanatiy.Lab3Java.primitives.MyVector;

public class Task4 implements Runnable {

private int startIndex;

private int endIndex;

private DataControl dataMonitor;

private TaskControl taskMonitor;

private int alpha4;

private int beta4;

private MyVector S4;

private MyMatrix MK4;

public Task4(DataControl dm, TaskControl tm) {

dataMonitor = dm;

taskMonitor = tm;

startIndex = Main.H \* 3;

endIndex = Main.N;

}

@Override

public void run() {

//input

Main.C.fill();

Main.E.fill();

dataMonitor.inputS();

taskMonitor.signalInputDone();

taskMonitor.waitInput();

//Calculate alpha

int alpha\_i = 0;

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

alpha\_i += Main.B.get(i) \* Main.C.get(i);

}

dataMonitor.calculateAlpha(alpha\_i);

taskMonitor.signalCalcAlphaDone();

//Calculate beta

int beta\_i = Integer.MAX\_VALUE;

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

if(Main.Z.get(i) < beta\_i)

beta\_i = Main.Z.get(i);

}

dataMonitor.calculateBeta(beta\_i);

taskMonitor.signalCalcBetaDone();

taskMonitor.waitCalcAlpha();

taskMonitor.waitCalcBeta();

//Copy data

alpha4 = dataMonitor.copyAlpha();

beta4 = dataMonitor.copyBeta();

S4 = dataMonitor.copyS();

MK4 = dataMonitor.copyMK();

//Calculate

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

//MO\*MK

MyVector temp = new MyVector(Main.N);

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

int element = 0;

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

element += Main.MO.get(i, k) \* MK4.get(k, j);

}

temp.set(j, element);

}

//S\*(MO\*MK)

int sum = 0;

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

sum += S4.get(j) \* temp.get(j);

}

//b\*S\*(MO\*MK)

sum \*= beta4;

//a\*E+b\*S\*(MO\*MK)

sum = alpha4 \* Main.E.get(i) + sum;

Main.A.set(i, sum);

}

//End calculate

taskMonitor.signalCalcDone();

}

}