НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА АВТОМАТИЗАЦІЇ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА УПРАВЛІННЯ

Лабораторна робота №6

з дисципліни **«**Паралельне програмування**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІП-31

Кахерський О.І.

Перевірив:

Корочкін О. В.

Київ – 2016 р.

**Тема:** Програмування для комп’ютерних систем зі спільною пам’яттю. Бібліотека OpenMP.

**Технічне завдання:** Розробити програму для розв’язання ПКС із СП (структура на рис. 1) математичної задачі: A =

**Бібліотека: OpenMP.**

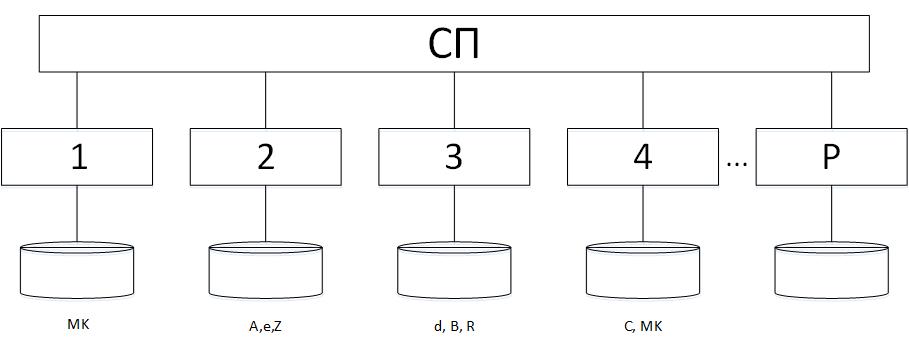


Рис.1 Структурна схема ПКС

**Виконання роботи**

**Етап 1.** Побудова паралельного алгоритму

2. =

Спільний ресурс: d, e, x, MK, R

**Етап 2. Розробка алогоритмів роботи кожного процесу**

i = (5 .. n), tID – номер поточної задачі, j = i/tID.

**Задача Т1**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід MK |  |
| 2. | Надіслати сигнал Т2..n про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т2..4 |  |
| 4. | Обчислити |  |
| 5. | Обчислити x = x + x1 | КД |
| 6. | Надіслати сигнал про завершення обчислення a задачам Т2..n |  |
| 7. | Очікувати сигнал про завершення обчислення x в задачах T2..n |  |
| 8. | Копіювати , e1 = e, MK1 = MK, х1 = х, R1 = R | КД |
| 9. | Виконати обчислення = |  |
| 10. | Надіслати сигнал Т2 про завершення обчислень |  |

**Задача Т2**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід e,Z |  |
| 2. | Надіслати сигнал Т1,3,4..n про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т1,3,4 |  |
| 4. | Обчислити |  |
| 5. | Обчислити х = х+х2 | КД |
| 6. | Надіслати сигнал про завершення обчислення х задачам Т1,3..n |  |
| 7. | Очікувати завершення обчислення x в задачах Т1,3..n | , |
| 8. | Копіювати , e2 = e, MK2 = MK, x2 = x, R2 = R | КД |
| 9. | Виконати обчислення = |  |
| 10. | Очікування завершення обчислення в Т1,3,4..n |  |
| 11. | Вивід А |  |

**Задача Т3**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід d,B,R |  |
| 2. | Надіслати сигнал Т1,2,4..n про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т1,2,4 |  |
| 4. | Обчислити |  |
| 5. | Обчислити х = х+х3 | КД |
| 6. | Надіслати сигнал про завершення обчислення х задачам Т1,2,4..n |  |
| 7. | Очікувати завершення обчислення х в інших задачах |  |
| 8. | Копіювати , e3 = e, MK3 = MK, х3 = х, R3 = R | КД |
| 9. | Виконати обчислення = |  |
| 10. | Надіслати сигнал Т2 про завершення обчислень |  |

**Задача Т4**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід C, MO |  |
| 2. | Надіслати сигнал Т1,2,3,i про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т1..3 |  |
| 4. | Обчислити |  |
| 5. | Обчислити х = х+х4 | КД |
| 6. | Надіслати сигнал про завершення обчислення х задачам Т1,2,3,i |  |
| 7. | Очікувати завершення обчислення х в інших задачах |  |
| 8. | Копіювати , e4 = e, MK4= MK, x4 = x, R4 = R | КД |
| 9. | Виконати обчислення = |  |
| 10. | Надіслати сигнал Т2 про завершення обчислень |  |

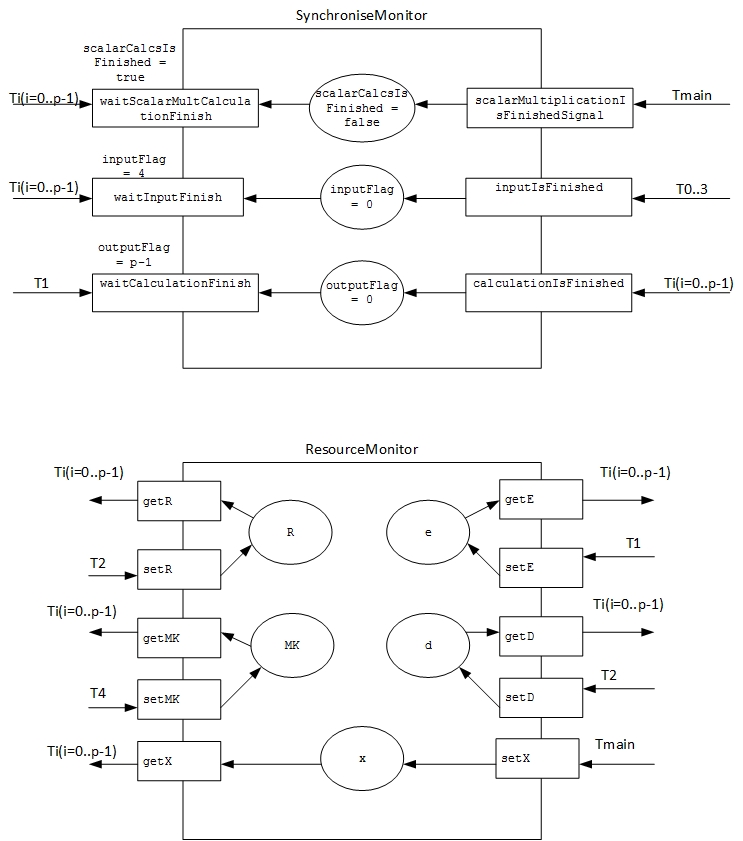
**Задача Тi**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Очікувати сигнал про завершення вводу в Т1..4 |  |
| 2. | Обчислити |  |
| 3. | Обчислити х = х+хi | КД |
| 4. | Надіслати сигнал про завершення обчислення х задачам Т1,2,3,j |  |
| 5. | Очікувати завершення обчислення х в інших задачах |  |
| 6. | Копіювати , ei = e, MKi= MK, xi= x, Ri = R | КД |
| 7. | Виконати обчислення = |  |
| 8. | Надіслати сигнал Т2 про завершення обчислень |  |

**Етап 3. Розробка структурної схеми взаємодії задач**

На структурній схемі взаємодії задач введено такі умовні позначення:

1. SynchroniseMonitor – клас, що реалізує монітор для вирішення завдання синхронізації;
2. ResourceMonitor – клас, що реалізує монітор для вирішення завдання синхронізації



**Етап 4. Лістинг коду**

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.ForkJoinPool;

/\*\*

\* Created by Oleh Kakherskyi, student of the KPI, FICT, IP-31 group (olehkakherskiy@gmail.com) on 19.04.2016.

\*/

**public class Main {**

public static int[] B, C, Z, A;

public static int[][] MO;

public static void main(String[] args) {

System.out.println("program is started");

Worker.N = Integer.parseInt(args[0]);

A = new int[Worker.N];

Worker.P = Integer.parseInt(args[1]);

SummTask.granular = Integer.parseInt(args[2]);

Worker.H = Worker.N / Worker.P;

ForkJoinPool forkJoinPool = new ForkJoinPool(Worker.P);

ResourceMonitor resourceMonitor = new ResourceMonitor();

SynchroniseMonitor synchroniseMonitor = new SynchroniseMonitor(Worker.P);

ExecutorService service = Executors.newFixedThreadPool(Worker.P);

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

service.execute(new Worker(synchroniseMonitor, resourceMonitor, i));

}

try {

synchroniseMonitor.waitInputFinish();

resourceMonitor.setX(forkJoinPool.invoke(new SummTask(B, C, 0, B.length)));

System.out.println("x = " + resourceMonitor.getX());

synchroniseMonitor.scalarMultiplicationIsFinishedSignal();

forkJoinPool.shutdown();

synchroniseMonitor.waitCalculationFinish();

service.shutdown();

System.out.println("program is finished");

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

import java.util.concurrent.RecursiveTask;

/\*\*

\* Created by Oleh Kakherskyi, student of the KPI, FICT, IP-31 group (olehkakherskiy@gmail.com) on 20.04.2016.

\*/

**public class SummTask extends RecursiveTask<Integer> {**

private int[] a;

private int[] b;

private int startIndex;

private int endIndex;

public static int granular;

public SummTask(int[] a, int[] b, int startIndex, int endIndex) {

this.a = a;

this.b = b;

this.startIndex = startIndex;

this.endIndex = endIndex;

}

@Override

protected Integer compute() {

if (granular >= endIndex - startIndex) {

return getScalarMultiplicationResult();

} else {

int mid = (startIndex + endIndex) / 2;

SummTask subtask1 = new SummTask(a, b, startIndex, mid);

subtask1.fork();

SummTask subtask2 = new SummTask(a, b, mid, endIndex);

subtask2.fork();

return subtask1.join() + subtask2.join();

}

}

private int getScalarMultiplicationResult() {

int result = 0;

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

result += a[i] \* b[i];

}

return result;

}

}

import java.util.Arrays;

/\*\*

\* Created by Oleh Kakherskyi, student of the KPI, FICT, IP-31 group (olehkakherskiy@gmail.com) on 20.04.2016.

\*/

**public class Worker implements Runnable {**

public static int N, P, H;

private SynchroniseMonitor synchMonitor;

private ResourceMonitor resourceMonitor;

private int tid;

public Worker(SynchroniseMonitor synchMonitor, ResourceMonitor resourseMonitor, int tid) {

this.synchMonitor = synchMonitor;

this.resourceMonitor = resourseMonitor;

this.tid = tid;

}

@Override

public void run() {

int startIndex = tid \* H;

int endIndex = tid == P - 1 ? N : (tid + 1) \* H;

//ввод данных

switch (tid) {

case 0: {

resourceMonitor.setMK(MatrixOperations.inputMatrix(N));

synchMonitor.inputIsFinished();

break;

}

case 1: {

resourceMonitor.setE(MatrixOperations.inputConstant());

Main.Z = MatrixOperations.inputVector(N);

synchMonitor.inputIsFinished();

break;

}

case 2: {

resourceMonitor.setD(MatrixOperations.inputConstant());

resourceMonitor.setR(MatrixOperations.inputVector(N));

Main.B = MatrixOperations.inputVector(N);

synchMonitor.inputIsFinished();

break;

}

case 3: {

Main.C = MatrixOperations.inputVector(N);

Main.MO = MatrixOperations.inputMatrix(N);

synchMonitor.inputIsFinished();

break;

}

}

try {

synchMonitor.waitInputFinish();

synchMonitor.waitScalarMultCalculationFinish();

int[][] MKi = resourceMonitor.getMK();

int[] Ri = resourceMonitor.getR();

int ei = resourceMonitor.getE();

int di = resourceMonitor.getD();

int xi = resourceMonitor.getX();

MatrixOperations.addVectors(Main.Z, MatrixOperations.

multVectorMatrix(Ri, MatrixOperations.multMatrix(Main.MO, MKi, startIndex, endIndex)), Main.A, ei, di \* xi, startIndex, endIndex);

synchMonitor.calculationIsFinished();

if (tid == 1) {

synchMonitor.waitCalculationFinish();

if (N <= 30)

System.out.println(Arrays.toString(Main.A));

}

System.out.println(String.format("Thread %d is finished calculations", tid));

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

/\*\*

\* Created by Oleh Kakherskyi, student of the KPI, FICT, IP-31 group (olehkakherskiy@gmail.com) on 20.04.2016.

\*/

**public class SynchroniseMonitor {**

private int inputFlag;

private int outputFlag;

private int threadCount;

private boolean scalarCalcsIsFinished = false;

public SynchroniseMonitor(int threadCount) {

this.threadCount = threadCount;

}

public synchronized void waitInputFinish() throws InterruptedException {

if (inputFlag < 4)

wait();

}

public synchronized void waitCalculationFinish() throws InterruptedException {

if (outputFlag < threadCount - 1)

wait();

}

public synchronized void waitScalarMultCalculationFinish() throws InterruptedException {

if (!scalarCalcsIsFinished)

wait();

}

public synchronized void scalarMultiplicationIsFinishedSignal() {

scalarCalcsIsFinished = true;

notifyAll();

}

public synchronized void inputIsFinished() {

if (++inputFlag == 4)

notifyAll();

}

public synchronized void calculationIsFinished() {

if (++outputFlag == 4)

notifyAll();

}

}

/\*\*

\* Created by Oleh Kakherskyi, student of the KPI, FICT, IP-31 group (olehkakherskiy@gmail.com) on 20.04.2016.

\*/

**public class ResourceMonitor {**

private int e;

private int d;

private int x;

private int[] R;

private int[][] MK;

public synchronized int getE() {

return e;

}

public synchronized void setE(int e) {

this.e = e;

}

public synchronized int getD() {

return d;

}

public synchronized void setD(int d) {

this.d = d;

}

public synchronized int getX() {

return x;

}

public synchronized void setX(int x) {

this.x = x;

}

public synchronized int[] getR() {

return MatrixOperations.copyVector(R);

}

public synchronized void setR(int[] r) {

R = r;

}

public synchronized int[][] getMK() {

return MatrixOperations.copyMatrix(MK);

}

public void setMK(int[][] MK) {

this.MK = MK;

}