

Міністерство освіти і науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота № 2

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

на тему:

“ C#. Семафори, мютекси, події, критичні секції”

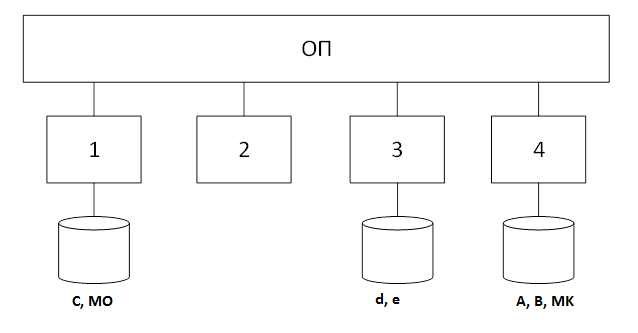
Виконав: студент ІІI курсу групи ІП – 42

Водотієць Денис Ігорович

Київ – 2017

**Технічне завдання**

Структура ПКС:



Математична задача: A = sort(d \* B + e \* C \* (MO \* MK)).

Мова та бібліотека програмування: C#.

**Виконання**

Етап 1. Розробка паралельного математичного алгоритму

1. A*H* = sort(d \* B*H* + e \* C \* (MO \* MK*H*));

2. A*2H* = mergeSort(A*H*, A*H*);

3. A = mergeSort(A*2H*, A*2H*).

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

A*H* - H елементів вектора А,

MK*H* - H рядків матриці МK,

mergeSort – сортування злиттям.

ЗР: e, d, C, MO.

Етап 2. Розробка алгоритму потоків

Потік Т1:

|  |  |
| --- | --- |
| 1. Введення C, MO; |  |
| 2. Сигнал Т2, Т3, Т4 про завершення введення; | S2,3,4:1 |
| 3. Очікувати сигналу завершення введення від Т3, Т4; | W3,4:1 |
| 4. Копіювати d1 := d, e1 = e, C1 := C, MO1 := MO; | КУ |
| 5. Рахунок1: A*H* = sort(d1 \* B*H* + e1 \* C1 \* (MO1 \* MK*H*)); |  |
| 6. Очікувати завершення рахунку від Т2; | W2:1 |
| 7. Рахунок2: A*2H* = mergeSort(A*H*, A*H*); |  |
| 8. Очікувати завершення злиття від Т3; | W3:2 |
| 9. Рахунок3: A = mergeSort(A*2H*, A*2H*); |  |
| 10. Сигнал Т4 про завершення обчислень. | S4:2 |

Потік Т2:

|  |  |
| --- | --- |
| 1. Очікувати сигналу завершення введення від Т1, T3, Т4; | W1,3,4:1 |
| 2. Копіювати d2 := d, e2 = e, C2 := C, MO2 := MO; | КУ |
| 3. Рахувати: A*H* = sort(d2 \* B*H* + e2 \* C2 \* (MO2 \* MK*H*)); |  |
| 4. Сигнал Т1 про завершення рахунку; | S1:1 |

Потік Т3:

|  |  |
| --- | --- |
| 1. Введення d, e; |  |
| 2. Сигнал Т1, Т2, Т4 про завершення введення; | S1,2,4:1 |
| 3. Очікувати сигналу завершення введення від Т1, Т4; | W1,4:1 |
| 4. Копіювати d3 := d, e3 = e, C3 := C, MO3 := MO; | КУ |
| 5. Рахувати: A*H* = sort(d3 \* B*H* + e3 \* C3 \* (MO3 \* MK*H*)); |  |
| 6. Очікувати завершення рахунку від Т4; | W4:2 |
| 7. Злиття A*2H* = mergeSort(A*H*, A*H*); |  |
| 8. Сигнал Т1 про завершення обчислень. | S1:2 |

Потік Т4:

|  |  |
| --- | --- |
| 1. Введення В, MK; |  |
| 2. Сигнал Т1, Т2, Т3 про завершення введення; | S1,2,3:1 |
| 3. Очікувати сигналу завершення введення від Т1, T3; | W1,3:1 |
| 4. Копіювати d4 := d, e4 = e, C4 := C, MO4 := MO; | КУ |
| 5. Рахувати: A*H* = sort(d4 \* B*H* + e4 \* C4 \* (MO4 \* MK*H*)); |  |
| 6. Сигнал Т3 про завершення рахунку; | S3:2 |
| 7. Очікувати завершення обрахунків від Т1; | W1:2 |
| 9. Вивести А. |  |

Етап 3. Розробка схеми взаємодії потоків



Етап 4. Розробка програми

using System;

using System.Threading;

/////////////////////////////////////////////////////////////

///// Lab 2. C# /////

///// A = sort(d\*B + e\*C\*(MO\*MK)) /////

///// Denys Vodotiiets /////

///// IP-42 /////

/////////////////////////////////////////////////////////////

namespace ConsoleApplication1

{

class Program

{

public static int N = 6;

public static int P = 4;

public static int H = N / P;

public static Matrix MO, MK;

public static Vector A, B, C;

public static int d, e;

public static Semaphore S1, S2, S3, S4;

public static Mutex MUT = new Mutex(false);

public static Object CS1 = new Object();

public static EventWaitHandle E\_InputT1;

public static EventWaitHandle E\_InputT3;

public static EventWaitHandle E\_InputT4;

static void Main(string[] args)

{

/\*if (N % P == 0)

{

H = N / P;

}

else

{

H = (N / P)+1;

}

Console.WriteLine(">>H = {0}", H);\*/

A = new Vector(N);

E\_InputT1 = new ManualResetEvent(false);

E\_InputT3 = new ManualResetEvent(false);

E\_InputT4 = new ManualResetEvent(false);

S1 = new Semaphore(0, 1);

S2 = new Semaphore(0, 1);

S3 = new Semaphore(0, 1);

S4 = new Semaphore(0, 1);

Thread t1 = new Thread(T1);

Thread t2 = new Thread(T2);

Thread t3 = new Thread(T3);

Thread t4 = new Thread(T4);

t1.Start();

t2.Start();

t3.Start();

t4.Start();

Console.ReadKey();

}

static void T1()

{

int k = 1;

int d1, e1;

Vector C1;

Matrix MO1;

Console.WriteLine(">>T1: Started");

//Input data

C = new Vector(N);

C.inputVector();

MO = new Matrix(N);

MO.inputMatrix();

//End input data

//Signal: end entering data

E\_InputT1.Set();

//Wait input data from T3 and T4

E\_InputT3.WaitOne();

E\_InputT4.WaitOne();

//Copy data

lock (CS1)

{

d1 = d;

e1 = e;

}

MUT.WaitOne();

C1 = new Vector(C);

MO1 = new Matrix(MO);

MUT.ReleaseMutex();

//End copy

//Calculating

Vector Temp = new Vector(N);

for (int i = (k - 1) \* H; i < k \* H; i++)

{

//MO1\*MK

for (int j = 0; j < N; j++)

{

int element = 0;

for (int l = 0; l < N; l++)

{

element += MK.getElement(i, l) \* MO1.getElement(l, j);

}

Temp.setElement(j, element);

}

//C1\*(MO1\*MK)

int sum = 0;

for (int j = 0; j < N; j++)

{

sum += Temp.getElement(j) \* C1.getElement(j);

}

//d1\*B + e1\*C1\*(MO1\*MK)

sum = e1 \* sum + B.getElement(i) \* d1;

A.setElement(i, sum);

}

A.sort((k - 1) \* H, H);

//End calculating

//Wait signal from T2

S1.WaitOne();

//Merge A2H

Vector tv = new Vector(2 \* H);

int point1 = (k - 1) \* H;

int point2 = k \* H;

for (int i = 0; i < tv.getLength(); i++)

{

if ((point2 >= (k + 1) \* H) || ((A.getElement(point1) <= A.getElement(point2)) && (point1 < k \* H)))

{

tv.setElement(i, A.getElement(point1));

point1++;

}

else

{

tv.setElement(i, A.getElement(point2));

point2++;

}

}

for (int i = 0; i < tv.getLength(); i++)

{

A.setElement((k - 1) \* H + i, tv.getElement(i));

}

//End merge

//Wait signal from T3

S3.WaitOne();

//Merge A

tv = new Vector(N);

point1 = 0;

point2 = 2 \* H;

for (int i = 0; i < tv.getLength(); i++)

{

if ((point2 >= N) || ((A.getElement(point1) <= A.getElement(point2)) && (point1 < 2 \* H)))

{

tv.setElement(i, A.getElement(point1));

point1++;

}

else

{

tv.setElement(i, A.getElement(point2));

point2++;

}

}

for (int i = 0; i < tv.getLength(); i++)

{

A.setElement((k - 1) \* H + i, tv.getElement(i));

}

//End merge

//Send signal T4

S4.Release();

Console.WriteLine(">>T1: Finished");

}

static void T2()

{

int k = 2;

int d2, e2;

Vector C2;

Matrix MO2;

Console.WriteLine(">>T2: Started");

//Wait input data from T1, T3, T4

E\_InputT1.WaitOne();

E\_InputT3.WaitOne();

E\_InputT4.WaitOne();

//Copy data

lock (CS1)

{

e2 = e;

d2 = d;

}

MUT.WaitOne();

C2 = new Vector(C);

MO2 = new Matrix(MO);

MUT.ReleaseMutex();

//End copy

//Calculating

Vector Temp = new Vector(N);

for (int i = (k - 1) \* H; i < k \* H; i++)

{

//MO2\*MK

for (int j = 0; j < N; j++)

{

int element = 0;

for (int l = 0; l < N; l++)

{

element += MK.getElement(i, l) \* MO2.getElement(l, j);

}

Temp.setElement(j, element);

}

//C2\*(MO2\*MK)

int sum = 0;

for (int j = 0; j < N; j++)

{

sum += Temp.getElement(j) \* C2.getElement(j);

}

//d2\*B + e2\*C2\*(MO2\*MK)

sum = e2 \* sum + B.getElement(i) \* d2;

A.setElement(i, sum);

}

A.sort((k - 1) \* H, H);

//End calculating

//Send signal T1

S1.Release();

Console.WriteLine(">>T2: Finished");

}

static void T3()

{

int k = 3;

int d3, e3;

Vector C3;

Matrix MO3;

Console.WriteLine(">>T3: Started");

//Input data

e = d = 1;

//End input data

//Signal: end entering data

E\_InputT3.Set();

//Wait input data from T1 and T4

E\_InputT1.WaitOne();

E\_InputT4.WaitOne();

//Copy data

lock (CS1)

{

e3 = e;

d3 = d;

}

MUT.WaitOne();

C3 = new Vector(C);

MO3 = new Matrix(MO);

MUT.ReleaseMutex();

//End copy

//Calculating

Vector Temp = new Vector(N);

for (int i = (k - 1) \* H; i < k \* H; i++)

{

//MO3\*MK

for (int j = 0; j < N; j++)

{

int element = 0;

for (int l = 0; l < N; l++)

{

element += MK.getElement(i, l) \* MO3.getElement(l, j);

}

Temp.setElement(j, element);

}

//C3\*(MO3\*MK)

int sum = 0;

for (int j = 0; j < N; j++)

{

sum += Temp.getElement(j) \* C3.getElement(j);

}

//d3\*B + e3\*C3\*(MO3\*MK)

sum = e3 \* sum + B.getElement(i) \* d3;

A.setElement(i, sum);

}

A.sort((k - 1) \* H, H);

//End calculating

//Wait signal from T4

S2.WaitOne();

//Merge A2H

Vector tv = new Vector(2 \* H);

int point1 = (k - 1) \* H;

int point2 = k \* H;

for (int i = 0; i < tv.getLength(); i++)

{

if ((point2 >= (k + 1) \* H) || ((A.getElement(point1) <= A.getElement(point2)) && (point1 < k \* H)))

{

tv.setElement(i, A.getElement(point1));

point1++;

}

else

{

tv.setElement(i, A.getElement(point2));

point2++;

}

}

for (int i = 0; i < tv.getLength(); i++)

{

A.setElement((k - 1) \* H + i, tv.getElement(i));

}

//End merge

//Send signal T1

S3.Release();

Console.WriteLine(">>T3: Finished");

}

static void T4()

{

int k = 4;

int d4, e4;

Vector C4;

Matrix MO4;

Console.WriteLine(">>T4: Started");

//Input data

B = new Vector(N);

B.inputVector();

MK = new Matrix(N);

MK.inputMatrix();

//End input data

//Signal: end entering data

E\_InputT4.Set();

//Wait signal from T1 and T3

E\_InputT1.WaitOne();

E\_InputT3.WaitOne();

//Copy data

lock (CS1)

{

e4 = e;

d4 = d;

}

MUT.WaitOne();

C4 = new Vector(C);

MO4 = new Matrix(MO);

MUT.ReleaseMutex();

//End copy

//Calculating

Vector Temp = new Vector(N);

for (int i = (k - 1) \* H; i < N; i++)

{

//MO4\*MK

for (int j = 0; j < N; j++)

{

int element = 0;

for (int l = 0; l < N; l++)

{

element += MK.getElement(l, i) \* MO4.getElement(j, l);

}

Temp.setElement(j, element);

}

//C4\*(MO4\*MK)

int sum = 0;

for (int j = 0; j < N; j++)

{

sum += Temp.getElement(j) \* C4.getElement(j);

}

//d4\*B + e4\*C4\*(MO4\*MK)

sum = e4 \* sum + B.getElement(i) \* d4;

A.setElement(i, sum);

}

A.sort((k - 1) \* H, N - (k - 1) \* H);

//End calculating

//Send signal T3

S2.Release();

//Wait signal from T1

S4.WaitOne();

Console.WriteLine("Calculation end.\nResult:");

if (N <= 100)

{

Console.WriteLine(A.outputVector());

}

else

{

Console.WriteLine("Vector is too large.");

}

Console.WriteLine(">>T4: Finished");

}

}

}

using System;

namespace ConsoleApplication1

{

class Matrix

{

private Vector[] matr;

public Matrix(int len)

{

matr = new Vector[len];

}

public Matrix(Matrix mat)

{

matr = new Vector[mat.getLength()];

for (int i = 0; i < matr.Length; i++)

{

matr[i] = new Vector(mat.getString(i));

}

}

public Vector getString(int i)

{

return matr[i];

}

public int getElement(int i, int j)

{

return matr[i].getElement(j);

}

public void setElement(int i, int j, int func)

{

matr[i].setElement(j, func);

}

public void setString(int i, Vector func)

{

matr[i] = func;

}

public int getLength()

{

return matr.Length;

}

public void inputMatrix()

{

for (int i = 0; i < matr.Length; i++)

{

matr[i] = new Vector(matr.Length);

matr[i].inputVector();

}

}

public String outputMatrix()

{

String s = "";

for (int i = 0; i < matr.Length; i++)

{

s = s + matr[i].outputVector() + "\n";

}

return s;

}

}

}

using System;

namespace ConsoleApplication1

{

class Vector

{

private int[] vect;

public Vector(int length)

{

vect = new int[length];

}

public Vector(Vector vec)

{

vect = new int[vec.getLength()];

for (int i = 0; i < vec.getLength(); i++)

{

vect[i] = vec.getElement(i);

}

}

public int getLength()

{

return vect.Length;

}

public int getElement(int i)

{

return vect[i];

}

public void setElement(int i, int func)

{

vect[i] = func;

}

public void sort(int a, int b)

{

Array.Sort(vect, a, b);

}

public void inputVector()

{

for (int i = 0; i < vect.Length; i++)

{

vect[i] = 1;

}

}

public String outputVector()

{

String s = "";

for (int i = 0; i < vect.Length; i++)

{

s = s + " " + vect[i];

}

return s;

}

}

}