Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 4

з дисципліни «Програмування для паралельних комп'ютерних систем»

на тему «Програмування для комп’ютерних систем зі спільною

пам’яттю. OpenMP. Критичні секції, atomic змінні»

Виконала:

студентка 3 курсу ФІОТ

групи ІО-51

Поштацька К. В.

Залікова – 5115

ПЕРЕВІРИВ:

Доцент, к.т.н.

Корочкін О.В.

Київ – 2018

**Тема:** «Програмування для комп’ютерних систем зі спільною пам’яттю. OpenMP. Критичні секції, atomic змінні»

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

A = d\* \*(MO\*MK) – e\* Sort()

Мова програмування: C++.



**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. ZHc= Sort(ZH), H= N/P
2. Z2Hc = Sort\*( ZHc ,ZHc)
3. Z4Hc = Sort\*( Z2Hc ,Z2Hc)
4. Z = Sort\*( Z4Hc ,Z2Hc)
5. A = d\* \*(MOH\*MK) – e\* ZH

Спільні ресурси: d, B, MK, e

**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Задача Т1** | | Точки синхронізації |
| 1. | Уведення e, z | |  |
| 2. | **Сигнал** задачам Т2, T3, T4, T5, T6 про введення e та z | | -- S2/3/4/5/6, 1 |
| 3. | **Чекати** на уведення даних в задачі Т2, T4 | | -- W2/4, 1 |
| 4. | Обчислення ZHc = Sort(ZH) | |  |
| 5. | **Чекати** на завершення сортування ZHc в задачі Т2 | | -- W2,2 |
| 6. | Z2Hc = Sort\*( ZHc ,ZHc) | |  |
| 7. | **Чекати** на завершення сортування Z2Hc в задачі Т3 | | -- W3/5,1 |
| 8. | Z = Sort\*( Z2Hc ,Z2Hc,Z2Hc) | |  |
| 9. | **Сигнал** задачам Т2, T3, T4, Т5, Т6 про завершення сортування Z | | -- S2/3/4/5/6, 2 |
| 10. | Копіювати d1 =d | | -- КД |
| 11. | Копіювати B1=B | | -- КД |
| 12. | Копіювати MK1=MK | | -- КД |
| 13. | Копіювати e1=e | | -- КД |
| 14. | Обчислення AH = d1\* \*(MOH\*MK1) – e1\* ZH | |  |
| 15. | **Чекати** завершення обчислень AH в Т2 | | -- W2,3 |
| 16. | **Чекати** завершення обчислень AH в T3, Т4, Т5 | | -- W3/4/5, 2 |
| 17. | **Чекати** завершення обчислень AH в T6 | | -- W6, 1 |
| 18. | Виведення результату А | |  |
|  | |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т2** | Точки синхронізації |
| 1. | Уведення B, MK |  |
| 2. | **Сигнал** Т1, Т3, Т4, T5, T6 про введення В, МК | -- S1/3/4/5/6, 1 |
| 3. | **Чекати** на уведення даних в задачі Т1, T4 | -- W1/4,1 |
| 4. | Обчислення ZH = Sort(ZH) |  |
| 5. | **Сигнал** задачі Т1 про завершення обчислень ZH | -- S1,2 |
| 6. | **Чекати** завершення обчислень Zв Т1 | -- W1,2 |
| 7. | Копіювати d2 =d | -- КД |
| 8. | Копіювати B2=B | -- КД |
| 9. | Копіювати MK2=MK | -- КД |
| 10. | Копіювати e2=e | -- КД |
| 11. | Обчислення AH = d2\* \*(MOH\*MK2) – e2\* ZH |  |
| 12. | **Сигнал** задачі Т1 про завершення обчислень MAH | -- S1,3 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т3** | Точки синхронізації |
| 1. | **Чекати** на уведення даних в задачі Т1, T2, T4 | -- W1/2/4,1 |
| 2. | Обчислення ZH = Sort(ZH) | -- КД |
| 3. | **Чекати** на завершення сортування ZHc в задачі Т4 | -- W4,2 |
| 4. | Z2Hc = Sort\*( ZHc ,ZHc) |  |
| 5. | **Сигнал** задачі Т1 про завершення сортування Z2Hc | -- S1,1 |
| 6. | **Чекати** завершення обчислень Zв Т1 | -- W1,2 |
| 7. | Копіювати d3 =d | -- КД |
| 8. | Копіювати B3=B | -- КД |
| 9. | Копіювати MK3=MK | -- КД |
| 10. | Копіювати e3=e | -- КД |
| 11. | Обчислення AH = d3\* \*(MOH\*MK3) – e3\* ZH |  |
| 12. | **Сигнал** задачі Т1 про завершення обчислень AH | -- S1,2 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т4** | Точки синхронізації |
| 1. | Уведення d, MO |  |
| 2. | **Сигнал** задачам Т1, T2, T3, T4, T5, T6 про введення d, MO | -- S1/2/3/5/6, 1 |
| 3. | **Чекати** на уведення даних в задачі Т1, T2 | -- W1/2,1 |
| 4. | Обчислення ZH = Sort(ZH) |  |
| 5. | **Сигнал** задачі T3 про завершення обчислень ZH | -- S3,2 |
| 6. | **Чекати** завершення обчислень Zв Т1 | -- W1,2 |
| 7. | Копіювати d4 =d | -- КД |
| 8. | Копіювати B4=B | -- КД |
| 9. | Копіювати MK4=MK | -- КД |
| 10. | Копіювати e4=e | -- КД |
| 11. | Обчислення AH = d4\* \*(MOH\*MK4) – e4\* ZH |  |
| 12. | **Сигнал** задачі Т1 про завершення обчислень AH | -- S1,2 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т5** | Точки синхронізації |
| 1. | **Чекати** на уведення даних в задачі Т1, T2, T4 | -- W1/2/4,1 |
| 2. | Обчислення ZH = Sort(ZH) | -- КД |
| 3. | **Чекати** на завершення сортування ZHc в задачі Т6 | -- W6,1 |
| 4. | Z2Hc = Sort\*( ZHc ,ZHc) |  |
| 5. | **Сигнал** задачі Т1 про завершення сортування Z2Hc | -- S1,1 |
| 6. | **Чекати** завершення обчислень Zв Т1 | -- W1,2 |
| 7. | Копіювати d5 =d | -- КД |
| 8. | Копіювати B5=B | -- КД |
| 9. | Копіювати MK5=MK | -- КД |
| 10. | Копіювати e5=e | -- КД |
| 11. | Обчислення AH = d5\* \*(MOH\*MK5) – e5\* ZH |  |
| 12. | **Сигнал** задачі Т1 про завершення обчислень AH | -- S1,2 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т6** | Точки синхронізації |
| 1. | **Чекати** на уведення даних в задачі Т1, T2, T4 | -- W1/2/4,1 |
| 2. | Обчислення ZH = Sort(ZH) |  |
| 3. | **Сигнал** задачі T5 про завершення обчислень ZH | -- S5,1 |
| 4. | **Чекати** завершення обчислень Zв Т1 | -- W1,2 |
| 5. | Копіювати d6 =d | -- КД |
| 6. | Копіювати B6=B | -- КД |
| 7. | Копіювати MK6=MK | -- КД |
| 8. | Копіювати e6=e | -- КД |
| 9. | Обчислення AH = d6\* \*(MOH\*MK6) – e6\* ZH |  |
| 10. | **Сигнал** задачі Т1 про завершення обчислень AH | -- S1,1 |

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

На структурній схемі взаємодії задач (рис. 2) задіяні такі бар’єри:

* Бар’єр 1 – для синхронізації з завершення вводу даних в T1, T2, T4;
* Бар’єр 2– для синхронізації з завершення сортування ZH T2, T4, T6;
* Бар’єр 3– сигнал задачі Т1 про завершення обчислень Z2Hc в Т3;
* Бар’єр 4– сигнал задачі Т1 про завершення обчислень Z2Hc в Т5;
* Бар’єр 5 – сигнал задачам T2, T3, T4, T5, T6 про завершення сортування Z в Т1;

Критичні секції: СS1 - для керування доступом до спільного ресурсу B;

CS2 - для керування доступом до спільного ресурсу MK;

d, e – atomic змінні.

**Етап 4. Розробка програми**

/\*

Laboratory work

on the topic "Threads in OpenMP library"

Variant: A = d\*B\*(MO\*MK)-e\*sort(Z)

Author: Poshtatska Katerina IO-51

\*/

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include <cstdlib>

#include <omp.h>

using namespace std;

int n = 12000; // size

int p = 6; //processors count

int H = n / p; //subsize

int d;

int e;

int\* A = new int[n];

int\* B = new int[n];

int\* Z = new int[n];

int\*\* MO = new int\*[n];

int\*\* MK = new int\*[n];

int\*\* MR = new int\*[n];

//Main method of Lab4

int main(void)

{

cout << "Lab4 started" << endl;

for (int i = 0; i < n; i++)

{

MR[i] = new int[n];

}

omp\_set\_num\_threads(p);

#pragma omp parallel shared (d,e,A,B,Z,MO,MK,MR)

{

cout << "TF"<< omp\_get\_thread\_num() <<" started" << endl;

//Input variables

if (omp\_get\_thread\_num() == 0) {

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

Z[i] = 1;

}

Z[0] = -1;

Z[1] = 2;

Z[2] = 1;

Z[3] = -3;

e = 1;

}

else if (omp\_get\_thread\_num() == 1) {

for (int i = 0; i < n; i++)

{

B[i] = 1;

MK[i] = new int[n];

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

{

MK[i][j] = 1;

}

}

}

else if (omp\_get\_thread\_num() == 3) {

for (int i = 0; i < n; i++)

{

MO[i] = new int[n];

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

{

MO[i][j] = 1;

}

}

d = 1;

}

//Input sync

#pragma omp barrier

//Sorting Zh

int\* X = new int[n];

int num = omp\_get\_thread\_num();

bool Finished;

int Temp = 0;

#pragma omp parallel

for (int i = 0; i < H; i++)

{

Finished = true;

#pragma omp parallel

for (int j = 1; j < (H - i); j++)

{

if (Z[j - 1 + H\* num] > Z[j + H \* num])

{

Temp = Z[j - 1 + H \* num];

Z[j - 1 + H \* num] = Z[j + H \* num];

Z[j + H \* num] = Temp;

Finished = false;

}

}

if (Finished)

{

break;

}

}

//Sorting Z2h

#pragma omp barrier

if (num % 2 == 0) {

int num\_T = 0;

int\* C = new int[2]{ num \* H, (num+1) \* H };

int\* C1 = new int[2]{ num \* H, (num + 1) \* H };

int\* C2 = new int[2]{ H, H };

#pragma omp parallel

for (int i = num \* H; i < n; i++)

{

Temp = INT\_MAX;

for (int j = 0; j < 2; j++)

{

if ((C[j] - C1[j]) < C2[j])

{

if (Z[C[j]] < Temp)

{

num\_T = j;

Temp = Z[C[j]];

}

}

}

X[i] = Temp;

C[num\_T]++;

}

#pragma omp parallel

for (int i = num \* H; i < (num+2)\*H; i++)

{

Z[i] = X[i];

}

}

//Sorting Zh = Sort(Z2h,Z2h,Z2h)

#pragma omp barrier

if (num == 0) {

int num\_T = 0;

int\*C = new int[3]{ 0, 2 \* H, 4\*H };

int\* C1 = new int[3]{ 0, 2 \* H, 4 \* H };

int\* C2 = new int[3]{ 2 \* H, 2 \* H, 2 \* H };

#pragma omp parallel

for (int i = 0; i < n; i++)

{

Temp = INT\_MAX;

#pragma omp parallel

for (int j = 0; j < 3; j++)

{

if ((C[j] - C1[j]) < C2[j])

{

if (Z[C[j]] < Temp)

{

num\_T = j;

Temp = Z[C[j]];

}

}

}

X[i] = Temp;

C[num\_T]++;

}

#pragma omp parallel

for (int i = 0; i < n; i++)

{

Z[i] = X[i];

}

}

//Sort sync

#pragma omp barrier

int di=0;

int ei=0;

int\* Bi;

int\*\* MKi;

//Copying resources

#pragma omp atomic

di += d;

#pragma omp atomic

ei += e;

#pragma omp critical(CS1)

{

Bi = B;

}

#pragma omp critical(CS2)

{

MKi = MK;

}

//Calculating Ah

#pragma omp parallel

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

{

#pragma omp parallel

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

{

MR[i][j] = 0;

#pragma omp parallel

for (int k = 0; k < n; k++)

{

MR[i][j] += MO[i][k] \* MKi[k][j];

}

MR[i][j] \*= di;

}

}

#pragma omp parallel

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

{

A[i] = 0;

#pragma omp parallel

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

{

A[i] += MR[i][j] \* Bi[j];

}

A[i] -= ei \* Z[i];

}

//Output sync

#pragma omp barrier

if (omp\_get\_thread\_num()==0)

{

if (n < 36) {

cout << "A = ";

for (int i = 0; i < n; i++)

{

cout << A[i] << " ";

}

cout << endl;

}

}

cout << "TF" << omp\_get\_thread\_num() << " finished" << endl;

}

cout << "Lab4 finished" << endl;

cin >> d;

}

