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 «OpenMP. Бар’єри, критичні секції»

ВИКОНАВ:

студент ІІІ курсу ФІОТ

групи ІО-51

Обозний Д.М.

Залікова – 5112

ПЕРЕВІРИВ:

Доцент, к.т.н.

Корочкін О.В.

Київ – 2018

**Тема:** «OpenMP. Бар’єри, критичні секції»

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

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

Мова програмування: C++. P = 4.

Засоби організації взаємодії процесів: Семафори, мютекси, події, критичні секції .



Рис. 1 Структурна схема ПКС

**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. ZHC = Sort(ZH)
2. Z2HC = Sort\*(ZHC, ZHC)
3. Z= Sort\*(Z2HC, Z2HC)
4. AH = d\*C\*(MOH\*MK)+e\*ZH

H = N/P, де N – розмір вектора.

Спільні ресурси: d, C, MK, e

Sort\* - сортування злиттям.

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

|  |  |  |
| --- | --- | --- |
|  | **Потік *P1*** | Точки синхронізації |
| 1. | Введення С, Z |  |
| 2. | **Сигнал** задачам P2, P3, P4 про введення С, Z | -- S2/3/4,1 |
| 3. | **Чекати** на уведення данних в задачах P2, P4 | -- W2/4,1 |
| 4. | Сортування ZHC = Sort(ZH) |  |
| 5. | **Сигнал** задачі P3 про завершення сортування Z | -- S3,2 |
| 6. | **Чекати** завершення сортування ZHC в P2, P3, P4 | -- W2/3/4,2 |
| 7. | Сортування Z2HC = Sort\*(ZHC, ZHC) |  |
| 8. | **Чекати** завершення сортування Z2HC в P3 | -- W3,3 |
| 9. | Сортування Z= Sort\*( Z2HC, Z2HC) |  |
| 10. | **Сигнал** задачам P2, P3, P4 про завершення сортування Z | -- S2/3/4,3 |
| 11. | Копіювати di = d | -- КД |
| 12. | Копіювати ei = e | -- КД |
| 13. | Копіювати Ci = C | -- КД |
| 14. | Копіювати MKi = MK | -- КД |
| 15. | Обчислення AH = di\*Ci\*(MOH\*MKi)+ei\*ZH |  |
| 16. | **Чекати** завершення обчислень AH в P2, P3, P4 | -- W2/3/4,4 |
| 17. | Виведення результату A |  |

|  |  |  |
| --- | --- | --- |
|  | **Потік *P2*** | Точки синхронізації |
| 1. | Введення d, MO |  |
| 2. | **Сигнал** задачам P1, P3, P4 про введення d, MO | -- S1/3/4,1 |
| 3. | **Чекати** на уведення данних в задачах P1, P4 | -- W1/4,1 |
| 4. | Сортування ZHC = Sort(ZH) |  |
| 5. | **Сигнал** задачі P1, P3 про завершення сортування ZHC | -- S1/3,2 |
| 7. | **Чекати** завершення сортування Z в задачі P1 | -- W1,2 |
| 8. | Копіювати di = d | -- КД |
| 9. | Копіювати ei = e | -- КД |
| 10. | Копіювати Ci = C | -- КД |
| 11. | Копіювати MKi = MK | -- КД |
| 12. | Обчислення AH = di\*Ci\*(MOH\*MKi)+ei\*ZH |  |
| 13. | **Сигнал** задачі P1 про завершення обчислень AH | -- S1,3 |

|  |  |  |
| --- | --- | --- |
|  | **Потік *P3*** | Точки синхронізації |
| 1. | **Чекати** на уведення данних в задачах P1, P2, P4 | -- W1/2/4,1 |
| 2. | Сортування ZHC = Sort(ZH) |  |
| 3. | **Сигнал** задачі P1 про завершення сортування ZHC | -- S1,1 |
| 4. | **Чекати** завершення сортування ZHC в P1, P2, P4 | -- W1/2/4,2 |
| 5. | Сортування Z2HC = Sort\*(ZHC, ZHC) |  |
| 6. | **Сигнал** задачі P1 про завершення сортування Z2HC | -- S1,2 |
| 7. | **Чекати** завершення сортування Z в задачі P1 | -- W1,3 |
| 8. | Копіювати di = d | -- КД |
| 9. | Копіювати ei = e | -- КД |
| 10. | Копіювати Ci = C | -- КД |
| 11. | Копіювати MKi = MK | -- КД |
| 12. | Обчислення AH = di\*Ci\*(MOH\*MKi)+ei\*ZH |  |
| 13. | **Сигнал** задачі P1 про завершення обчислень AH | -- S1,3 |

|  |  |  |
| --- | --- | --- |
|  | **Потік *P4*** | Точки синхронізації |
| 1. | Введення e, MK |  |
| 2. | **Сигнал** задачам P1, P2, P3 про введення e, MK | -- S1/2/3,1 |
| 3. | **Чекати** на уведення данних в задачах P1, P2 | -- W1/2,1 |
| 4. | Сортування ZHC = Sort(ZH) |  |
| 5. | **Сигнал** задачі P1, P3 про завершення cортування ZHC | -- S1/3,2 |
| 7. | **Чекати** завершення злиття Z в задачі P1 | -- W1,2 |
| 8. | Копіювати di = d | -- КД |
| 9. | Копіювати ei = e | -- КД |
| 10. | Копіювати Ci = C | -- КД |
| 11. | Копіювати MKi = MK | -- КД |
| 12. | Обчислення AH = di\*Ci\*(MOH\*MKi)+ei\*ZH |  |
| 13. | **Сигнал** задачі P1 про завершення обчислень AH | -- S1,3 |

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

Взаємодія потоків Р1, Р2, P3,P4 пов’язана з вирішення наступних завдань:

* Синхронізації по введенню даних (*Z* та *C* в потоці *Р1*, *d* та *MO* в потоці *Р2* і *e* та *MK* в потоці *P4*);
* Синхронізації по завершенню сортування підвектора *ZH* (Потоки *Р1, P3* чекають на завершення сортування в інших потоках);
* Синхронізації по завершенню сортування підвектора *Z2H* (Потік *Р1* чекають на завершення сортування в потоці *P3*);
* Синхронізації по завершенню сортування вектора *Z* (Потоки *Р2*, *Р3*, *Р4* чекають на завершення сортування в потоці *Р1*);
* Синхронізації по завершенню обчислень вектора *АH* (Потік *Р1* чекає на завершення обчислень в потоках *Р2*, *Р3*, *Р4*);
* Завдання взаємного виключення при роботі потоків із спільними ресурсами (змінні *d*, *e*, *C*, *MK*);

Для вирішення завдання взаємного виключення використаємо механізм критичних секцій та atomic змінних, для синхронізації – бар’єри.

d та e – atomic змінні



Схема взаємодії потоків

**Етап 4. Розробка програми**

/\*

Laboratory work

on the topic "Threads in OpenMP library"

Variant: A = d\*C\*(MO\*MK)+e\*sort(Z)

Created: 06.03.2018 8:52 PM

Author: Obozniy Dmitriy IO-51

\*/

#include "stdafx.h"

#include "Data.h"

#include <windows.h>

#include <iostream>

#include <cstdlib>

#include <omp.h>

using namespace std;

int n = 1000; // size

int p = 4; //processors count

int h = n / p; //subsize

int d;

int e;

int\* A = new int[n];

int\* C = new int[n];

int\* Z = new int[n];

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

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

//Main method of Lab4

int main(void)

{

cout << "Lab4 started" << endl;

/\*

Variable n using for rhe size of all matrix and vectors in program.

data is an exeplar of class Data with parametr n.

\*/

Data(n,p);

omp\_set\_num\_threads(p);

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

{

//Input variables

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

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

if (n < 1)

{

C = Vector\_Input("C");

Z = Vector\_Input("Z");

}

else {

C = Vector\_Input1(1);

Z = Vector\_Input1(1);

}

Z[5] = -1;

Z[2] = 0;

Z[9] = 2;

}

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

if (n < 1)

{

cout << "d = ";

cin >> d;

MO = Matrix\_Input("MO");

}

else {

d = 1;

MO = Matrix\_Input1(1);

}

}

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

if (n < 1)

{

cout << "e = ";

cin >> e;

MK = Matrix\_Input("MK");

}

else {

e = 1;

MK = Matrix\_Input1(1);

}

}

//Input sync

#pragma omp barrier

//Sorting Zh

Func1(Z, omp\_get\_thread\_num());

//Sorting Zh sync

#pragma omp barrier

//Sorting Z2h

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

Vector\_Together(Z, omp\_get\_thread\_num(),0);

}

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

Vector\_Together(Z, omp\_get\_thread\_num(), 0);

}

//Splitting Z sync

#pragma omp barrier

//Splitting Z

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

Vector\_Together(Z, omp\_get\_thread\_num(), 1);

}

#pragma omp barrier

int di=0;

int ei=0;

int\* Ci;

int\*\* MKi;

//Copying resources

#pragma omp atomic

di += d;

#pragma omp atomic

ei += e;

#pragma omp critical(CS1)

{

Ci = C;

}

#pragma omp critical(CS2)

{

MKi = MK;

}

//Calculating Ah

Func2(MO, MKi, A, Ci, Z, di, ei, omp\_get\_thread\_num());

//Output sync

#pragma omp barrier

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

{

if (n <= 20)

{

cout << "A= ";

Vector\_Output(A);

}

}

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

}

cout << "Lab4 finished" << endl;

cin >> d;

}