Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №4**

**з дисципліни «Програмування паралельних комп’ютерних систем»**

**на тему «OpenMP»**

Виконав:

студент ІІI курсу

групи ІО-53

Лісовий В. О.

Перевірив:

доц. Корочкін О. В.

Київ – 2018

**Тема**

«Програмування для комп’ютерних систем зі спільною пам’яттю. OpenMP»

**Технічне завдання**

Розробити програму для розв’язання ПКС із СП (структура на рис. 1) математичної задачі: A = (B \* C) \* S + max(T) \* Z \* (MO \* MH).

Мова програмування: C/C++, бібліотека OpenMP.

Засоби взаємодії: бар’єри, критичні секції, замки.



Рис. 1 Структурна схема ПКС

**Виконання роботи**

**Етап 1. Побудова паралельного алгоритму**

1. sc\_li = (BH \* CH), i = 1..4;
2. sc\_g = ;
3. max\_li = max(TH), i = 1..4;
4. max\_g = max(max\_g, max\_li), i = 1..4;
5. AH = sc\_g \* SH + max\_g \* Z \* MO \* MHH;

Спільний ресурс: sc\_g, max\_g, Z, MO.

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

|  |  |  |
| --- | --- | --- |
| № | Задача Т1 |  |
| 1 | Ввід даних (B) |  |
| 2 | Сигнал усім потокам про завершення вводу даних | S\*,1 |
| 3 | Чекати завершення вводу даних в Т2, Т3, Т4 | W\*,1 |
| 4 | Обчислення sc\_l= BH \* CH |  |
| 5 | Обчислення sc\_g= sc\_g + sc\_l | КД |
| 6 | Обчислення max\_l= max(TH) |  |
| 7 | Обчислення max\_g = max(max\_g, max\_l) | КД |
| 8 | Копіювання sc\_l = sc\_g, max\_l = max\_g, Z1 = Z, MO1 = MO | КД |
| 9 | Сигнал усім потокам про завершення обчислення скаляру та пошуку максимума | S\*,2 |
| 10 | Чекати завершення завершення обчислення скаляру та пошуку максимума в Т2, Т3, Т4 | W\*,2 |
| 11 | Обчислення AH = sc\_g \* SH + max\_g \* Z \* MO \* MHH |  |
| 12 | Сигнал усім потокам про завершення обчислення | S\*,3 |
| 13 | Чекати завершення завершення обчислення в Т2, Т3, Т4 | W\*,3 |
| 14 | Вивід А |  |

|  |  |  |
| --- | --- | --- |
| № | Задача Т2 |  |
| 1 | Ввід даних (С, MH) |  |
| 2 | Сигнал усім потокам про завершення вводу даних | S\*,1 |
| 3 | Чекати завершення вводу даних в Т1, Т3, Т4 | W\*,1 |
| 4 | Обчислення sc\_l= BH \* CH |  |
| 5 | Обчислення sc\_g= sc\_g + sc\_l | КД |
| 6 | Обчислення max\_l= max(TH) |  |
| 7 | Обчислення max\_g = max(max\_g, max\_l) | КД |
| 8 | Копіювання sc\_l = sc\_g, max\_l = max\_g, Z2 = Z, MO2 = MO | КД |
| 9 | Сигнал усім потокам про завершення обчислення скаляру та пошуку максимума | S\*,2 |
| 10 | Чекати завершення завершення обчислення скаляру та пошуку максимума в Т1, Т3, Т4 | W\*,2 |
| 11 | Обчислення AH = sc\_g \* SH + max\_g \* Z \* MO \* MHH |  |
| 12 | Сигнал усім потокам про завершення обчислення | S\*,3 |

|  |  |  |
| --- | --- | --- |
| № | Задача Т3 |  |
| 1 | Ввід даних (S, MO) |  |
| 2 | Сигнал усім потокам про завершення вводу даних | S\*,1 |
| 3 | Чекати завершення вводу даних в Т1, Т2, Т4 | W\*,1 |
| 4 | Обчислення sc\_l= BH \* CH |  |
| 5 | Обчислення sc\_g= sc\_g + sc\_l | КД |
| 6 | Обчислення max\_l= max(TH) |  |
| 7 | Обчислення max\_g = max(max\_g, max\_l) | КД |
| 8 | Копіювання sc\_l = sc\_g, max\_l = max\_g, Z3 = Z, MO3 = MO | КД |
| 9 | Сигнал усім потокам про завершення обчислення скаляру та пошуку максимума | S\*,2 |
| 10 | Чекати завершення завершення обчислення скаляру та пошуку максимума в Т1, Т2, Т4 | W\*,2 |
| 11 | Обчислення AH = sc\_g \* SH + max\_g \* Z \* MO \* MHH |  |
| 12 | Сигнал усім потокам про завершення обчислення | S\*,3 |

|  |  |  |
| --- | --- | --- |
| № | Задача Т4 |  |
| 1 | Ввід даних (T, Z) |  |
| 2 | Сигнал усім потокам про завершення вводу даних | S\*,1 |
| 3 | Чекати завершення вводу даних в Т1, Т2, Т3 | W\*,1 |
| 4 | Обчислення sc\_l= BH \* CH |  |
| 5 | Обчислення sc\_g= sc\_g + sc\_l | КД |
| 6 | Обчислення max\_l= max(TH) |  |
| 7 | Обчислення max\_g = max(max\_g, max\_l) | КД |
| 8 | Копіювання sc\_l = sc\_g, max\_l = max\_g, Z4 = Z, MO4 = MO | КД |
| 9 | Сигнал усім потокам про завершення обчислення скаляру та пошуку максимума | S\*,2 |
| 10 | Чекати завершення завершення обчислення скаляру та пошуку максимума в Т1, Т2, Т3 | W\*,2 |
| 11 | Обчислення AH = sc\_g \* SH + max\_g \* Z \* MO \* MHH |  |
| 12 | Сигнал усім потокам про завершення обчислення | S\*,3 |

**Етап 3. Розробка програми**

#include "functions.h"

#include <omp.h>

using namespace std;

int n, p = 4 , num = 1;

float h;

int \*A, \*B, \*C, \*S, \*T, \*Z, \*MO, \*MH;

int sc\_g = 0 , max\_g = INT32\_MIN;

void mem\_alloc();

void input(int tid);

void mem\_free();

void F(int \*A, int sc, int \*S, int max, int \*Z, int \*MO, int \*MH, int from, int to);

omp\_lock\_t copy\_lock;

int main()

{

cout << "Enter size:" << endl;

cin >> n;

h = (float) n / p;

mem\_alloc();

omp\_init\_lock(&copy\_lock);// init lock

#pragma omp parallel num\_threads(p)

{

const int tid = omp\_get\_thread\_num();

cout << "Thread " << tid << " start" << endl;

input(tid);

#pragma omp barrier // wait input

int sc\_l;

int max\_l;

int \*Z\_l, \*MO\_l;

int leftIndex = (int)(tid \* h);

int rightIndex = (int)((tid + 1) \* h);

sc\_l = scalar(B, C, leftIndex, rightIndex);

#pragma omp critical(scalar)

{

sc\_g += sc\_l;

}

max\_l = max(T, leftIndex, rightIndex);

#pragma omp critical(max)

{

if (max\_g < max\_l)

max\_g = max\_l;

}

#pragma omp barrier // wait finish scalar and max

omp\_set\_lock(&copy\_lock);

sc\_l = sc\_g;

max\_l = max\_g;

Z\_l = Z;

MO\_l = MO;

omp\_unset\_lock(&copy\_lock);

F(A, sc\_l, S, max\_l, Z\_l, MO\_l, MH, leftIndex, rightIndex);

#pragma omp barrier // wait finish computing

if (tid == 0) {

cout << "A = ";

printVector(A, n);

}

cout << "Thread " << tid << " finish" << endl;

}

omp\_destroy\_lock(&copy\_lock);

mem\_free();

cout << "Press any key..." << endl;

int i;

cin >> i;

return 0;

}

void input(int tid) {

switch(tid) {

case 0:

fillArr(B, num, n);

return;

case 1:

fillArr(C, num, n);

fillArr(MH, num, n\*n);

return;

case 2:

fillArr(S, num, n);

fillArr(MO, num, n\*n);

return;

case 3:

fillArr(T, num, n);

fillArr(Z, num, n);

//T[2] = 2;

return;

}

}

void mem\_alloc() {

A = (int \*)malloc(n\*4);

B = (int \*)malloc(n\*4);

C = (int \*)malloc(n\*4);

S = (int \*)malloc(n\*4);

T = (int \*)malloc(n\*4);

Z = (int \*)malloc(n\*4);

MO = (int \*)malloc(n\*n\*4);

MH = (int \*)malloc(n\*n\*4);

}

void mem\_free() {

free(A);

free(B);

free(C);

free(S);

free(T);

free(Z);

free(MO);

free(MH);

}

void F(int \*A, int sc, int \*S, int max, int \*Z, int \*MO, int \*MH, int from, int to) {

int \*MBuf = (int \*)malloc(n \* n \* 4);

matrixMultiplication(MO, MH, MBuf, from, to, n);

vectorMatrixMultiplication(Z, MBuf, A, from, to, n);

for (int i = from; i < to; i++) {

A[i] = A[i] \* max + sc \* S[i];

}

free(MBuf);

}

#include "functions.h"

void printVector(int \*A, int n) {

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

printf("%d ", A[i]);

printf("\n");

}

void fillArr(int \*A, int num, int n) {

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

A[i] = num;

}

/\*\*

\* Multiplication matrix A and B

\* @param A - matrix

\* @param B - matrix

\* @param C=A\*B - matrix

\*/

void matrixMultiplication(int \*A, int \*B, int \*C, int from, int to, int n) {

int buf;

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

for (int j = from; j < to; j++) {

buf = 0;

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

buf += A[i\*n + k] \* B[k\*n + j];

}

C[i\*n + j] = buf;

}

}

}

/\*\*

\* Multiplication vector A with matrix B

\* @param A - vector

\* @param B - matrix

\* @param C=A\*B - vector

\*/

void vectorMatrixMultiplication(int \*A, int \*B, int \*C, int from, int to, int n) {

int buf;

for (int i = from; i < to; i++ ) {

buf = 0;

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

buf += A[j] \* B[j\*n + i];

}

C[i] = buf;

}

}

int scalar(int \*A, int \*B, int from, int to) {

int scalar = 0;

for (int i = from; i < to; i++)

scalar += A[i] \* B[i];

return scalar;

}

int max(int \*T, int from, int to) {

int max = INT32\_MIN;

for (int i = from; i < to; i++)

if (T[i] > max) {

max = T[i];

}

return max;

}