Національний технічний університет України

«Київський Політехнічний Інститут»

Факультет інформатики і обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №4**

**З предмету «Паралельні та розподілені обчислення»**

**OpenMP**

Виконав:

Студент  
IІІ курсу ФІОТ  
групи ІО-12  
Бута С. О.

Залікова книжка №1205

Київ-2014

**Техническое задание**

1. Структура ПКС с ОП: 
2. Задача: .
3. Язык программирования: C++; библиотека: OpenMP.
4. Средства взаимодействия задач: критические секции, атомарные операции, барьеры.

**Выполнение работы**

1. Разработка параллельного математического алгоритма.
2. AH = sort(B\*MOH); OP: B;
3. A2H = mergeSort(AH, AH);
4. A4H = mergeSort(A2H, A2H);
5. A3H = mergeSort(A2H, AH);
6. A = mergeSort(A4H, A3H);
7. AH = AH + l\*C\*(MK\*MTH); OP: l, C, MK;
8. Разработка алгоритмов процессов.

**Задача Т1:**

1. Ввод МТ, В
2. Синхронизация Тj (j=1..P) по вводу Sk;1 Wk;1 (k=1..P; k≠1)
3. Копия1 В КУ
   1. B1 = B
4. Счёт1
   1. AH = sort(B1\*MOH)
5. Синхронизация Тj (j=1..P) по Счёт1 Sk;2 Wk;2 (k=1..P; k≠1)
6. Счёт2
   1. A2H = mergeSort(AH, AH)
7. Синхронизация Тj (j=1..3) по Счёт2 S2,3;1 W2,3;1
8. Счёт3
   1. A4H = mergeSort(A2H, A2H)
9. Синхронизация Т1, T2 по Счёт3 и Счёт4 S2;1
10. Счёт5
    1. A = mergeSort(A4H, A3H)
11. Синхронизация Тj (j=1..P) по Счёт5 Sk;3 Wk;3 (k=1..P; k≠1)
12. Копии2 l, C, MK КУ
    1. l1 = l
    2. C1 = C
    3. MK1 = MK
13. Счёт6
    1. AH = AH + l\*C\*(MK\*MTH)
14. Синхронизация Тj (j=1..P) по Счёт6 Sk;4 Wk;4
15. Вывод А

**Задача Т2:**

1. Ввод МO
2. Синхронизация Тj (j=1..P) по вводу Sk;1 Wk;1 (k=1..P; k≠2)
3. Копия1 В КУ
   1. B2 = B
4. Счёт1
   1. AH = sort(B1\*MOH)
5. Синхронизация Тj (j=1..P) по Счёт1 Sk;2 Wk;2 (k=1..P; k≠2)
6. Счёт2
   1. A2H = mergeSort(AH, AH)
7. Синхронизация Тj (j=1..3) по Счёт2 S1,3;1 W1,3;1
8. Счёт4
   1. A3H = mergeSort(A2H, AH)
9. Синхронизация Т1, T2 по Счёт3 и Счёт4 S1;1
10. Синхронизация Тj (j=1..P) по Счёт5 Sk;3 Wk;3 (k=1..P; k≠2)
11. Копии2 l, C, MK КУ
    1. L2 = l
    2. C2 = C
    3. MK2 = MK
12. Счёт6
    1. AH = AH + l\*C\*(MK\*MTH)
13. Синхронизация Тj (j=1..P) по Счёт6 Sk;4 Wk;4 (k=1..P; k≠2)

**Задача Т3:**

1. Ввод C
2. Синхронизация Тj (j=1..P) по вводу Sk;1 Wk;1 (k=1..P; k≠3)
3. Копия1 В КУ
   1. B3 = B
4. Счёт1
   1. AH = sort(B1\*MOH)
5. Синхронизация Тj (j=1..P) по Счёт1 Sk;2 Wk;2 (k=1..P; k≠3)
6. Счёт2
   1. A2H = mergeSort(AH, AH)
7. Синхронизация Тj (j=1..3) по Счёт2 S1,2;1 W1,2;1
8. Синхронизация Тj (j=1..P) по Счёт5 Sk;3 Wk;3 (k=1..P; k≠3)
9. Копии2 l, C, MK КУ
   1. L3 = l
   2. C3 = C
   3. MK3 = MK
10. Счёт6
    1. AH = AH + l\*C\*(MK\*MTH)
11. Синхронизация Тj (j=1..P) по Счёт6 Sk;4 Wk;4 (k=1..P; k≠3)

**Задача Т4:**

1. Ввод l, MK
2. Синхронизация Тj (j=1..P) по вводу Sk;1 Wk;1 (k=1..P; k≠4)
3. Копия1 В КУ
   1. B4 = B
4. Счёт1
   1. AH = sort(B1\*MOH)
5. Синхронизация Тj (j=1..P) по Счёт1 Sk;2 Wk;2 (k=1..P; k≠4)
6. Синхронизация Тj (j=1..P) по Счёт5 Sk;3 Wk;3 (k=1..P; k≠4)
7. Копии2 l, C, MK КУ
   1. L4 = l
   2. C4 = C
   3. MK4 = MK
8. Счёт6
   1. AH = AH + l\*C\*(MK\*MTH)
9. Синхронизация Тj (j=1..P) по Счёт6 Sk;4 Wk;4 (k=1..P; k≠4)

**Задача Тi (i = 5, 6, 7):**

1. Синхронизация Тj (j=1..P) по вводу Sk;1 Wk;1 (k=1..P; k≠i)
2. Копия1 В КУ
   1. Bi = B
3. Счёт1
   1. AH = sort(B1\*MOH)
4. Синхронизация Тj (j=1..P) по Счёт1 Sk;2 Wk;2 (k=1..P; k≠i)
5. Синхронизация Тj (j=1..P) по Счёт5 Sk;3 Wk;3 (k=1..P; k≠i)
6. Копии2 l, C, MK КУ
   1. Li = l
   2. Ci = C
   3. MKi = MK
7. Счёт6
   1. AH = AH + l\*C\*(MK\*MTH)
8. Синхронизация Тj (j=1..P) по Счёт6 Sk;4 Wk;4 (k=1..P; k≠4)

**Разработка программы.**

**Листинг**:

Исходный код Runner.cpp

#include <iostream>

#include <omp.h>

#include "Vector.h"

#include "Matrix.h"

using namespace std;

const int N = 1400;

const int P = 7;

const int H = N / P;

const int FILLER = 1;

int l;

Vector A(N), B(N), C(N);

Matrix MK(N), MO(N), MT(N);

void mergeSort(int from1, int to1, int from2, int to2, int copyPos){

Vector A1(to1 - from1 + 1);

A.getPart(A1, from1, to1);

Vector A2(to2 - from2 + 1);

A.getPart(A2, from2, to2);

Vector Ares(A1.getSize() + A2.getSize());

A1.mergeSort(Ares, A2);

Ares.copyFrom(A, copyPos);

}

int main(){

omp\_set\_dynamic(0); // запретить библиотеке openmp менять число потоков во время исполнения

omp\_set\_num\_threads(P); // установить число потоков в P

#pragma omp parallel shared(l, A, B, C, MK, MO, MT)

{

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

#pragma omp critical(Start)

cout<< "Thread" << thread+1 << " started.\n";

#pragma omp sections

{

//Ввод МТ, В

#pragma omp section

{

MT.input();

B.input();

}

//Ввод МO

#pragma omp section

{

MO.input();

//MO[0][2] = 2; //проверка правильности сортировки

}

//Ввод C

#pragma omp section

{

C.input();

}

//Ввод l, MK

#pragma omp section

{

l = FILLER;

MK.input();

}

}

//Синхронизация Тj (j=1..P) по вводу

#pragma omp barrier

//Копия1 В

Vector Bi(N);

#pragma omp critical(Copy1)

{

B.copy(Bi);

}

//Счёт1

Vector res(H);

Bi.mult(res, MO, thread\*H, thread\*H + H - 1);

res.sort();

res.copyFrom(A, thread\*H);

//Синхронизация Тj (j=1..P) по Счёт1

#pragma omp barrier

//Счёт2

#pragma omp sections

{

#pragma omp section

{

mergeSort(0, H-1, H, 2\*H-1, 0);

}

#pragma omp section

{

mergeSort(2\*H, 3\*H-1, 3\*H, 4\*H-1, 2\*H);

}

#pragma omp section

{

mergeSort(4\*H, 5\*H-1, 5\*H, 6\*H-1, 4\*H);

}

}

//Синхронизация Тj (j=1..3) по Счёт2

#pragma omp barrier

#pragma omp sections

{

//Счёт3

#pragma omp section

{

mergeSort(0, 2\*H-1, 2\*H, 4\*H-1, 0);

}

//Счёт4

#pragma omp section

{

mergeSort(4\*H, 6\*H-1, 6\*H, 7\*H-1, 4\*H);

}

}

//Синхронизация Т1, T2 по Счёт3 и Счёт4

#pragma omp barrier

//Счёт5

#pragma omp sections

{

#pragma omp section

{

mergeSort(0, 4\*H-1, 4\*H, 7\*H-1, 0);

}

}

//Синхронизация Тj (j=1..P) по Счёт5

#pragma omp barrier

//Копии2 l, C, MK

int li;

Vector Ci(N);

Matrix MKi(N);

#pragma omp critical(Copy2)

{

li = l;

C.copy(Ci);

MK.copy(MKi);

}

//Счёт6

Vector v(H);

Matrix Mres(N);

MK.mult(Mres, MT, thread\*H, thread\*H+H-1);

Ci.mult(v, Mres, thread\*H, thread\*H+H-1);

v.mult(li);

A.add(v, thread\*H);

//Синхронизация Тj (j=1..P) по Счёт6

#pragma omp barrier

//Вывод А

#pragma omp sections

{

#pragma omp section

{

cout << "Vector A: \n";

for (int i=0; i < A.getSize(); i++){

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

}

cout << endl;

}

}

#pragma omp critical(Finish)

cout<< "Thread" << thread+1 << " finished.\n";

}

getchar();

return 0;

}