Лабораторная работа №5

В данной лабораторной работе использовались секции. Благодаря им я распараллелил код на 2 и 4 части(потока). Ограничивается количество секций только мощности железа. Синхронизация производилась за счет двух разных переменных ввиду того, что использование критических секций и даже atomic приводило к увеличению времени в 10 раз. И хотя код распаралеленый критическими секциями на 2 части выполнялся медленнее, чем через редукшены, но в коде, где необходимо выполнять 2 разных участка кода пралельно он даст достаточную производительность.

В результате было проверено без распаралеливания, с распаралеливанием на 2 потока, и критические секции на 2 и 4 части, массив из 150000000;

Результаты:

1.178s without paralel 7.50011e+12

0.579s with reduction 7.50011e+12

0.752s with 2 section 7.50011e+12

0.625s with 4 section 7.50011e+12

Код:

#include "header.h"

#include <algorithm>

#include <time.h>

using namespace std;

const int N = 150000000;

void **five**() {

//srand(time(NULL));

int i;

double composition2 = 0;

double timed2 = 0;

double finish, start;

int\* A = new int[N];

double\* B = new double[N];

double\* C = new double[N];

random\_device rd;

mt19937 mersenne(rd());

for (i = 0; i < N; i++)

{

A[i] = (mersenne() % 100000)+1;

B[i] = 1+(mersenne() % 100000);

C[i] = 1+(mersenne() % 100000);

}

start = omp\_get\_wtime();

#pragma omp parallel private(timed2) num\_threads(1)

{

#pragma omp for reduction(+:composition2)

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

{

if ((A[i] % 2) == 0)

{

timed2 = B[i] / C[i];

}

else

{

timed2 = B[i] + A[i];

}

composition2 += timed2;

}

}

finish = omp\_get\_wtime();

cout << finish - start << "s without paralel ";

cout << composition2 << endl;

composition2 = 0;

start = omp\_get\_wtime();

#pragma omp parallel private(timed2) num\_threads(2)

{

#pragma omp for reduction(+:composition2)

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

{

if ((A[i] % 2) == 0)

{

timed2 = B[i] / C[i];

}

else

{

timed2 = B[i] + A[i];

}

composition2 += timed2;

}

}

finish = omp\_get\_wtime();

cout << finish - start << "s with reduction ";

cout << composition2 << endl;

omp\_set\_num\_threads(12);

double\* sum = new double[2];

sum[0]=0;

sum[1]=0;

composition2 = 0;

start = omp\_get\_wtime();

#pragma omp parallel sections private(timed2) num\_threads(12)

{

#pragma omp section

{

for (int i = 0; i < N/2; i++)

{

if ((A[i] % 2) == 0)

{

timed2 = B[i] / C[i];

}

else

{

timed2 = B[i] + A[i];

}

sum[0] += timed2;

}

}

#pragma omp section

{

for (int i = N/2; i < N; i++)

{

if ((A[i] % 2) == 0)

{

timed2 = B[i] / C[i];

}

else

{

timed2 = B[i] + A[i];

}

sum[1] += timed2;

}

}

}

composition2 = sum[0] + sum[1];

finish = omp\_get\_wtime();

cout << finish - start << "s with 2 section ";

cout << composition2 << endl;

sum = new double[4];

composition2 = 0;

sum[0]=0;

sum[1]=0;

sum[2]=0;

sum[3]=0;

start = omp\_get\_wtime();

#pragma omp parallel sections private(timed2) num\_threads(12)

{

#pragma omp section

{

for (int i = 0; i < N/4; i++)

{

if ((A[i] % 2) == 0)

{

timed2 = B[i] / C[i];

}

else

{

timed2 = B[i] + A[i];

}

sum[0] += timed2;

}

}

#pragma omp section

{

for (int i = N/4; i < N/4\*2; i++)

{

if ((A[i] % 2) == 0)

{

timed2 = B[i] / C[i];

}

else

{

timed2 = B[i] + A[i];

}

sum[1] += timed2;

}

}

#pragma omp section

{

for (int i = N/4\*2; i < N/4\*3; i++)

{

if ((A[i] % 2) == 0)

{

timed2 = B[i] / C[i];

}

else

{

timed2 = B[i] + A[i];

}

sum[2] += timed2;

}

}

#pragma omp section

{

for (int i = N/4\*3; i < N; i++)

{

if ((A[i] % 2) == 0)

{

timed2 = B[i] / C[i];

}

else

{

timed2 = B[i] + A[i];

}

sum[3] += timed2;

}

}

}

composition2 = sum[0] + sum[1]+sum[2]+sum[3];

finish = omp\_get\_wtime();

cout << finish - start << "s with 4 section ";

cout << composition2 << endl;

system("pause");

}