Технологии параллельного программирования

Модуль 1. Введение. Технология OpenMP

Семинар 2

**Задание 1. Выполнить задачу умножения матриц с использованием директивы параллелизации OpenMP.**

**Сравнить время выполнения задач. Также сравнить со временем выполнения задач без параллелизации, из предыдущего задания.**

#include <omp.h>

#include <iostream>

#include <stdio.h>

#include <locale>

#include <time.h>

using namespace std;

// Функция выделения памяти под 2-ный массив

double\*\* malloc\_array(long int n)

{

double\*\* matrix = new double\* [n];

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

matrix[i] = new double[n];

return matrix;

}

// Функция освобождения памяти

void free\_array(double\*\* matrix, long int n)

{

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

delete[] matrix[i];

delete[] matrix;

}

// Функция инициализации матрицы случайными числами из [0,1]

void rand\_init\_matrix(double\*\* matrix, long int n)

{

srand(time(NULL));

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

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

matrix[i][j] = rand() / RAND\_MAX;

}

// Функция обнуления матрицы

void zero\_init\_matrix(double\*\* matrix, long int n)

{

srand(time(NULL));

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

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

matrix[i][j] = 0.0;

}

int main()

{

const long int N = 1000;

double\*\* A, \*\* B, \*\* C;

// Выделение памяти под матрицы A,B,C

A = malloc\_array(N);

B = malloc\_array(N);

C = malloc\_array(N);

// Инициализация матриц

rand\_init\_matrix(A, N);

rand\_init\_matrix(B, N);

zero\_init\_matrix(C, N);

clock\_t t;

// Перемножение матриц с порядком циклов jki

zero\_init\_matrix(C, N);

t = clock();

#pragma omp parallel for

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

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

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

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

t = clock() - t;

cout << "Время умножения jki цикла " << t / CLOCKS\_PER\_SEC << " seconds" << endl;

// Перемножение матриц с порядком циклов ikj

zero\_init\_matrix(C, N);

t = clock();

#pragma omp parallel for

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

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

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

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

t = clock() - t;

cout << "Время умножения ikj цикла " << t / CLOCKS\_PER\_SEC << " seconds" << endl;

// Освобождение памяти занимаемой матрицами A,B,C

free\_array(A, N);

free\_array(B, N);

free\_array(C, N);

system("pause");

return 0;

}

**Задание 2. Распараллеливание директивой OpenMP задач по числу ядер процессора / логических процессоров**

#include <stdio.h>

#include <locale>

int main()

{

setlocale(LC\_ALL, "Russian");

printf("Последовательная область 1\n");

#pragma omp parallel

{

printf("Параллельная область\n");

}

printf("Последовательная область 2\n");

}

**Задание 3. Пример использования директивы parallel с опцией shared**

#include <iostream>

#include <omp.h>

#include <string>

using namespace std;

int main()

{

string hw = "Hello World!\n";

#pragma omp parallel shared(hw)

{

cout << hw;

}

return 0;

}

**Задание 4. Пример программы с ситуацией «Гонка данных»**

#include <iostream>

#include "omp.h"

#include <string>

using namespace std;

int main()

{

int x = 0;

#pragma omp parallel shared(x) num\_threads(30)

{

x += 1;

}

cout << "x = " << x << endl;

return 0;

}