Технологии параллельного программирования

Модуль 1. Технология OpenMP

Семинар 8

Скомпилировать и выполнить программный код на языке С++. Полученные результаты представить в отчете (скринами) преподавателю, сделать краткие выводы.

**Задание 1. Векторизация с использованием SIMD intrinsic класс**

#include <dvec.h>

#include <iostream>

#define \_INC\_IOSTREAM 1

#define ALIGN \_\_attribute\_\_ ((aligned (32)))

using namespace std;

int main()

{

F64vec4 A[4], B[4], C[4];

int i, j;

for (i = 0, j = 0; i < 16; i += 4, j++)

{

A[j] = F64vec4(i, i + 1, i + 2, i + 3);

B[j] = F64vec4(i, i + 1, i + 2, i + 3);

}

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

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

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

{

for (int j = 0; j<4; j++)

{

cout << "C[" << i \* 4 + j << "]=" << C[i][3 - j] << endl;

}

}

return 0;

}

**Задание 2. Векторизация с использованием директивы OpenMP simd**

#include <iostream>

#include <chrono>

using namespace std::chrono;

const int n = 6;

using namespace std;

auto start = high\_resolution\_clock::now();

int main()

{

int i, j;

float\* a = new float[n\*n];

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

a[i] = 1.0f;

for (int k = 0; k < 1; k++)

{

for (int i = 1; i < 3; i++)

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

{

a[n\*i + j] = a[n\*i + j - (i + 1) - 1] / (a[n\*i + j - (i + 1)] \* a[n\*i + j - (i + 1)]) + 1;

}

for (int i = 3; i < n; i++)

#pragma omp simd safelen(4)

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

{

a[n\*i + j] = a[n\*i + j - (i + 1) - 1] / (a[n\*i + j - (i + 1)] \* a[n\*i + j - (i + 1)]) + 1;

}

}

auto stop = high\_resolution\_clock::now();

cout << "example2\_vec" << endl;

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

{

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

{

cout << a[i\*n + j] << " ";

}

cout << endl;

auto duration = duration\_cast<microseconds>(stop - start);

     cout << "Time taken by function: "

         << duration.count() << " microseconds" << endl;

}

return 0;

}