ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №1

“Матричні обчислення за допомогою OpenMP”

за курсом «Розподілені обчислення»

студента групи ПА-20-1

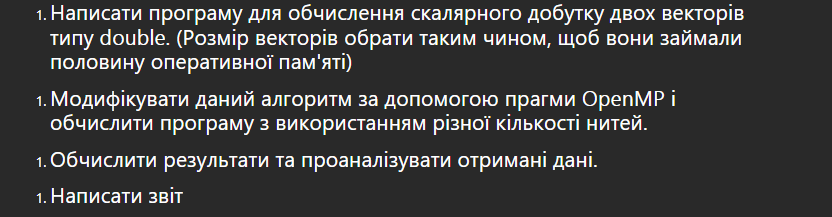
Афанасьєва Дениса Костянтиновича

кафедра комп’ютерних технологій, ДНУ

 2022/2023 н.р.

Зміст

1. **Постановка задачі**



1. **Опис розв’язку**

Для того щоб підключити бібліотеку OpenMP треба у властивостях проекту у вкладці «С/C++» у розділі «Мова» напроти параметру «Підтримка OpenMP» обрати пункт «Так», після чого підключити у самому файлі програми бібліотеку «omp.h». Та для використання можливостей бібліотеки треба використовувати відповідні директиви.

Далі написав програму для обрахування скалярного добутку 2 векторів та після перевірки її на працездатність почав модифікувати її, для цього використав директиву omp «for», яка розподіляє роботу циклу між заданою кількістю потоків.

Далі почав проводити аналіз роботи програми, і дізнався що чим більше використовується потоків, тим скоріше програма виконує задані дії, ось таблиця результатів:

|  |  |
| --- | --- |
| Кількість потоків | Час (сек.) |
| 1 | 2.886 |
| 2 | 1,546 |
| 3 | 1,124 |
| 4 | 0,996 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Потоки | Час у 5 спробах (сек) | | | | |
| 1 | 3,77 | 2,66 | 2,64 | 2,75 | 2,61 |
| 2 | 2,18 | 1,37 | 1,34 | 1,35 | 1,49 |
| 3 | 1,2 | 1,07 | 1,05 | 1,1 | 1,2 |
| 4 | 1,02 | 0,93 | 0,94 | 1,11 | 0,98 |

Для того, щоб добитися більш точного результату я робив 5 випробувань та обчислював середнє арифметичне часу.

З таблиці можна побачити закономірність часу від кількості потоків.

1. **Вихідний код програми**

#include <iostream>

#include <cmath>

#include <omp.h>

using namespace std;

#define NUM\_THREADS 1

int main() {

setlocale(LC\_ALL, "");

const int size = (int)pow(1024, 3);

int\* a = new int[size];

int\* b = new int[size];

int\* sum = new int[size];

double beg, end;

cout << "Start of initialization" << endl;

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

a[i] = 5;

b[i] = 5;

}

cout << "Initialization's end" << endl;

cout << "Number of threads " << NUM\_THREADS << endl;

omp\_set\_dynamic(0);

for(int j = 0; j<5;j++){

beg = omp\_get\_wtime();

#pragma omp parallel num\_threads(NUM\_THREADS)

{

#pragma omp for

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

a[i] = a[i] \* b[i];

}

}

end = omp\_get\_wtime();

double t = end - beg;

cout << j+1 << ") Time: " << t << endl;

}

}

1. **Приклади роботи програми**

