НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

з лабораторної роботи №2

з дисципліни «Розподілені і хмарні обчислення» на тему

*Основи OpenMP. Робота з матрицями*

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студенка групи КМ-81 | *Ст.викладач Ліскін В. О.* |
| *Верзун П.В.* |  |

Київ — 2021

# **ЗМІСТ**

[ВСТУП](#_4kzg94r1oe7u) 3

[1.ПОСТАНОВКА ЗАДАЧІ](#_oqnhgm2d969n) 4

[2.ОСНОВНА ЧАСТИНА](#_qimzfjvzy9a3) 5

[Вивід програми](#_klqj299s9ub7) 5

[3.ВИСНОВКИ](#_ba72zbw9ra6) 6

[ДОДАТОК 1. ТЕКСТ ПРОГРАМИ](#_hb0hboaqmyi0) 7

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **ВСТУП**

Для перевірки того, що компілятор підтримує якусь версію OpenMP, достатньо написати директиви умов компіляції #ifdef або #ifndef. Якщо деякий участок системи оточити викликами функції omp\_get\_wtime(), то різниця повертаємих значень покаже час роботи даного участку коду. Функція omp\_get\_wtick() повертає в деякому участку коду час у секундах на момент виклику. Цей час можна розглядати як точність системного таймеру.

# 

# **1.ПОСТАНОВКА ЗАДАЧІ**

**Завдання 1**

1. Визначте, яку версію OpenMP підтримує компілятор на доступній

системі.

2. За допомогою функцій OpenMP визначте час, потрібній системі для

роботу функції вимірювання час. Визначити точність системного

таймера.

3. Напишіть просту програму, яка виводить «Hello World» у паралельній

області.

**Завдання 2**

Реалізувати множення квадратних матриць з використанням OpenMP.

# 

# 

# 

# 

# 

# **2.ОСНОВНА ЧАСТИНА**

В цій лабораторній роботі було реалізовано функціонал виводу рядку у паралельній області. Було порівняно час виконання у паралельній та непаралельній областях. Також було реалізовано множення матриць також у двох випадках: паралельному та непаралельному. Програма була розроблена на C++ і за допомогою OpenMP

# **Вивід програми**

Можна побачити різницю у часі виконання і як паралельність пришвидшує роботу програми.



# **3.ВИСНОВКИ**

В лабораторній роботі було отримано базові знання про можливості OpenMP та було реалізовано програму, яка на прикладі виводу рядків та множенні матриць показує те, що паралельне виконання програми може значно пришвидшити її роботу.

# **ДОДАТОК 1. ТЕКСТ ПРОГРАМИ**

#include "omp.h"

#include <stdio.h>

#include <iostream>

#include <unordered\_map>

#define N 500

int main() {

std::unordered\_map<unsigned,std::string> map{

{200505,"2.5"},{200805,"3.0"},{201107,"3.1"},{201307,"4.0"},{201511,"4.5"},{201811,"5.0"},{202011,"5.1"}};

std::cout << "OpenMP found" << map.at(\_OPENMP) << ".\n";

double start, end, tick;

start = omp\_get\_wtime();

std::cout << "Non-parallel\n\n";

end = omp\_get\_wtime();

tick = omp\_get\_wtick();

std::cout << "Without parallelism: " << end - start << "\n\n";

std::cout << "Accuracy of timer: " << tick << "\n\n";

start = omp\_get\_wtime();

#pragma omp parallel num\_threads(4)

{

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

printf("Hello world \tThread:%d\n", i);

}

end = omp\_get\_wtime();

std::cout << "\nResult with threads: " << end - start;

std::cout << "\n\nMatrix task:" << std::endl;

double a[N][N], b[N][N], c[N][N];

int i, j, k, sum, sum\_2;

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

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

a[i][j] = b[i][j] = rand() % 10;

}

}

start = omp\_get\_wtime();

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

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

c[i][j] = 0.0;

for (k = 0; k < N; k++) c[i][j] += a[i][k] \* b[k][j];

}

}

end = omp\_get\_wtime();

std::cout << "\nResult without threads: " << end - start << "\n";

start = omp\_get\_wtime();

#pragma omp parallel for private(i, j, k)

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

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

c[i][j] = 0.0;

for (k = 0; k < N; k++) c[i][j] += a[i][k] \* b[k][j];

}

}

end = omp\_get\_wtime();

std::cout << "Result with parallel: " << end - start << "\n\n";

start = omp\_get\_wtime();

#pragma omp

#pragma omp parallel shared(a) private(i,j,sum)

#pragma omp for

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

sum =0;

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

sum +=a[i][j];

printf ("Summary of elems in row %d equals %d\n",i,sum);

} /\* Завершение параллельного фрагмента \*/

end = omp\_get\_wtime();

printf("Execution time: %f", end - start);

printf("\n\n");

start = omp\_get\_wtime();

int total = 0;

#pragma omp

#pragma omp parallel for shared(a) private(i,j,sum\_2) reduction (+:total)

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

sum\_2 = 0;

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

sum\_2 +=a[i][j];

}

printf ("Summary of elems in row %d equals %d\n",i,sum\_2);

total = total +sum\_2;

}

printf("Total summary:%d\n", total);

end = omp\_get\_wtime();

printf("Execution time: %f\n", end - start);

return 0;

}