МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет

«Харківський політехнічний інститут»

Навчально-науковий інститут комп’ютерного моделювання, прикладної фізики та математики

Кафедра динаміки та міцності машин

Звіт

про виконання

лабораторної роботи № 11

по курсу «Комп’ютерні мережі та розподілені обчислювання»

з теми "Паралельні та обчислювальні системи

(OMP)"

                                                    Виконав:

студент групи ІКМ-220Б

Пономаренко О.В.

Викладач:

доцент кафедри ДММ

Трубаєв О.І.

Харків 2022

**Варіант:** 12

**Завдання:**

1. Ознайомтесь з прикладом *omp\_integrate.c,* що знаходитьсяпоруч із цим файлом та реалізує інтегрування функції методом прямокутників. Скомпілюйте та запустіть його.
2. Модифікуйте код програми таким чином, щоб він реалізовував метод трапецій та підраховував інтеграл у діапазоні [-*k, k*] від функції *y(x) = k⋅xk*, де *k* – ваш номер варіанту (номер в списку у журналі). Обчисліть аналітичне значення інтегралу та порівняйте його з отриманим чисельно. Визначте похибку.
3. Визначте час та прискорення від роботи програми на 1, 2, 3, 4, 5, 6, 7, 8 потоках та при nsteps = 40 000 000 і 80 000 000. Проведіть досліди на вашому комп’ютері та обчислювальному кластері. Побудуйте графіки прискорення від кількості ядер (для цього використовуйте файл *template.xlsx*) (для кластера та вашого ПК).

**Код програми:**

#include <math.h>

#include <stdio.h>

#include <omp.h>

const double PI = 3.14159265358979323846;

const double k = 2;

const double a = -k;

const double b = k;

const int nsteps = 40000000;

double func(double x) {

return k \* pow(x, k);

}

double integrate(double(\*func)(double), double a, double b, int n) {

const double width = (b - a) / n;

double trapezoidal\_integral = 0;

for (int step = 0; step < n; step++) {

const double x1 = a + step \* width;

const double x2 = a + (step + 1) \* width;

trapezoidal\_integral += 0.5 \* (x2 - x1) \* (func(x1) + func(x2));

}

return trapezoidal\_integral;

}

double integrate\_omp(double(\*func)(double), double a, double b, int n) {

const double width = (b - a) / n;

double trapezoidal\_integral = 0;

#pragma omp parallel

{

int nthreads = omp\_get\_num\_threads();

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

int items\_per\_thread = n / nthreads;

int lb = threadid \* items\_per\_thread;

int ub = (threadid == nthreads - 1) ? (n - 1) : (lb + items\_per\_thread - 1);

double trapezoidal\_integral\_step = 0.0;

for (int step = lb; step <= ub; step++) {

const double x1 = a + step \* width;

const double x2 = a + (step + 1) \* width;

trapezoidal\_integral\_step += 0.5 \* (x2 - x1) \* (func(x1) + func(x2));

}

#pragma omp atomic

trapezoidal\_integral += trapezoidal\_integral\_step;

}

return trapezoidal\_integral;

}

double run\_serial() {

double t = omp\_get\_wtime();

double res = integrate(func, a, b, nsteps);

t = omp\_get\_wtime() - t;

printf("Result (serial): %.12f; error %.12f\n", res, fabs(res - 0));

return t;

}

double run\_parallel() {

double t = omp\_get\_wtime();

double res = integrate\_omp(func, a, b, nsteps);

t = omp\_get\_wtime() - t;

printf("Result (parallel): %.12f; error %.12f\n", res, fabs(res - 0));

return t;

}

int main(int argc, char\*\* argv) {

printf("Integration f(x) on [%.12f, %.12f], nsteps = %d\n", a, b, nsteps);

double tserial = run\_serial();

double tparallel = run\_parallel();

printf("Execution time (serial): %.6f\n", tserial);

printf("Execution time (parallel): %.6f\n", tparallel);

printf("Speedup: %.2f\n", tserial / tparallel);

return 0;

}

**Вимірювання:**

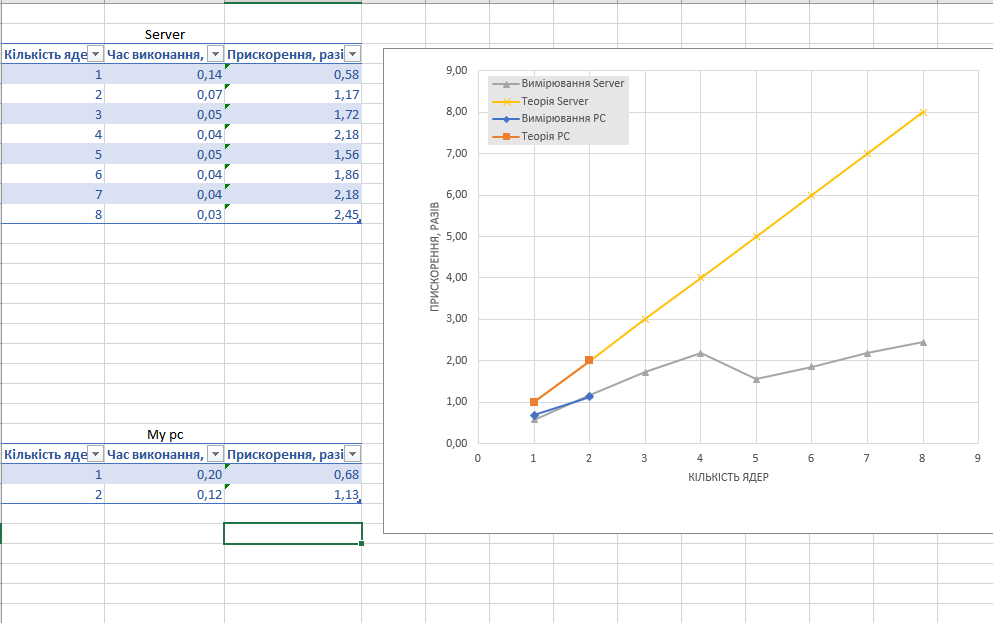


Рисунок 1 - 40 000 000 ітерацій

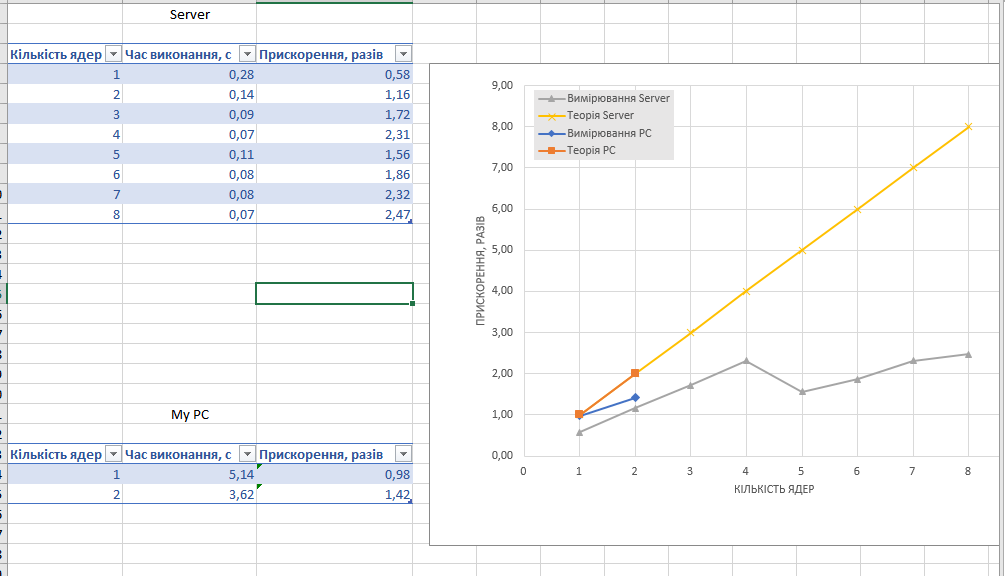
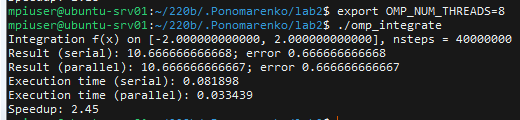
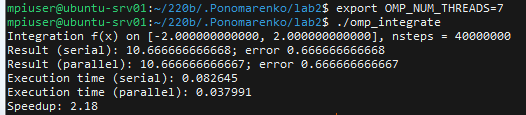
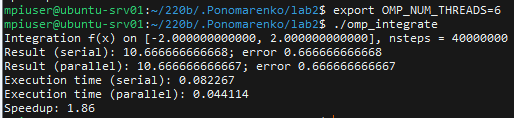
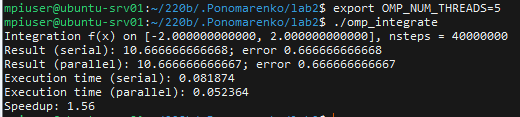
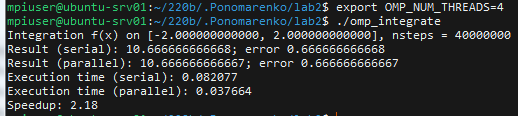
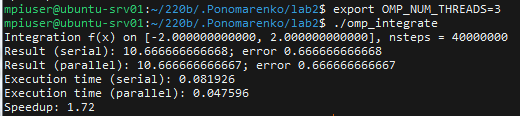
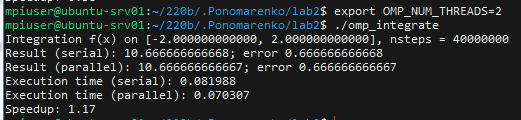
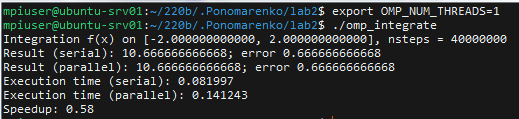


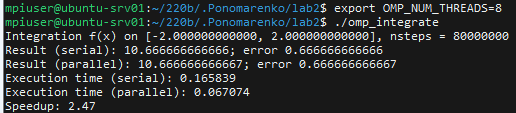
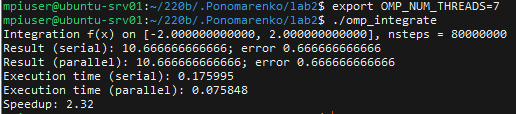
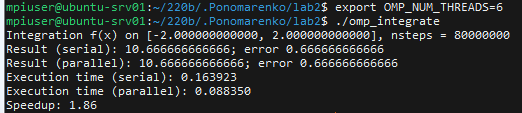
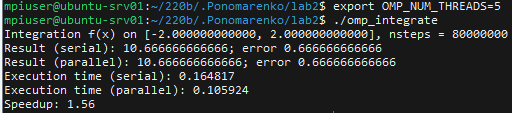
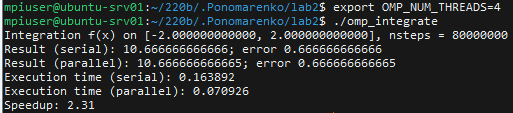
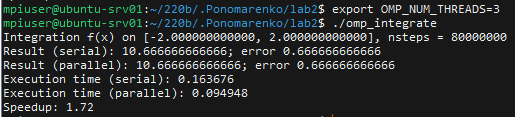
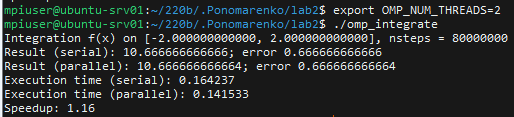
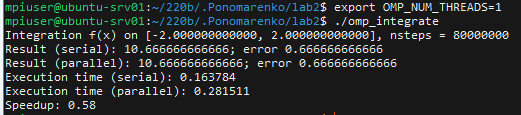
Рисунок 2 - 80 000 000 ітерацій

**Скріншоти роботи програми з різною кількістю потоків:**

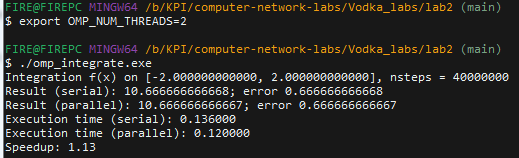
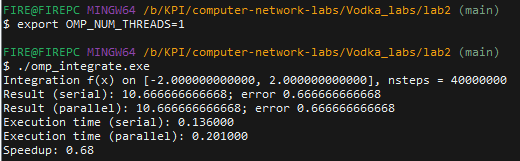
**Сервер 40 000 000 кроків:**

****

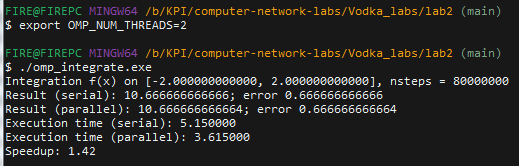
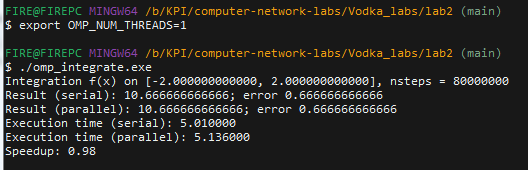
**Сервер 80 000 000 кроків:**

****

**ПК 40 000 000 кроків:**

****

**ПК 80 000 000 кроків:**

****