**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

Факультет прикладної математики та інформатики

Кафедра програмування

Звіт про виконання

лабораторної роботи №1

**Розпаралелення додавання/віднімання матриць**

з курсу “Паралельні та розподілені обчислення”

Виконав:

Студент групи ПМО-21с

Заяц Ростислав Васильович

Перевірив:

Гошко Богдан Мирославович

**Львів – 2025**

**Тема:** Написання програми додавання\віднімання матриць з використанням паралельного та послідовного програмування

**Мета:** Ознайомитись з паралельними алгоритмами програмування, навчитись використовувати їх, для оптимізації та пришвидшення програми.

**Завдання: «**Напишіть програми обчислення суми та різниці двох матриць (послідовний та паралельний алгоритми). Порахуйте час роботи кожної з програм, обчисліть прискорення та ефективність роботи паралельного алгоритму.

В матрицях розмірності (n,m) робіть змінними, щоб легко змінювати величину матриці.

Кількість потоків k - також змінна величина. Програма повинна показувати час при послідовному способі виконання програми, а також при розпаралеленні на k потоків.

Зверніть увагу на випадки, коли розмірність матриці не кратна кількості потоків.!!!»

**Прискорення (Speedup)**

Визначення:  
 Прискорення — це відношення часу виконання послідовного алгоритму до часу виконання паралельного алгоритму. Воно показує, наскільки швидше працює паралельна версія в порівнянні з послідовною.

Формула:

***Speedup = T\_посл / T\_парал***

Де:  
- T\_посл — час виконання послідовного алгоритму;  
- T\_парал — час виконання паралельного алгоритму.

**Ефективність (Efficiency)**

Визначення:  
 Ефективність показує, наскільки ефективно використовуються потоки (чи процесори) при паралельному виконанні. Це відношення прискорення до кількості потоків.

Формула:

***Efficiency = (Speedup / k) \* 100%***

Де:  
- Speedup — прискорення;  
- k — кількість потоків (процесорів).

**Хід роботи:**

Реалізовано звичайну програму для роботи з матрицями з використанням ООП. Перевизначено оператор додавання та віднімання для використання в поcлідовному алгоритмі, оператор доступу[], для реалізації паралельного.

**Послідовний алгоритм:**

Операції + та – ми можемо застосовувати тільки для матриць однакових розмірів, тому спочатку в операторах +\- ми перевіряємо чи збігаються розміри матриць, а потім виконуємо звичайне, поелементне додавання послідовно в одному потоці. В коді це виглядає так:

1. Ініціалізуємо 3 матриці (A,B,C) А,В – матриці над якими будемо виконувати операції, С – результативна матриця.
2. Засікаємо час за допомогою clock() з бібліотеки ctime
3. Обраховуємо час виконання.

**Паралельний алгоритм:**

Основна ідеї розбити послідовне додавання на k паралельно виконуючих обчислення потоків, що допоможе пришвидшити нашу програму. Розбиваємо ми наступним чином:

Ділимо кількість рядків матриці на кількість потоків, візьмемо до прикладу 100 / 4 = 25, число 25 це крок нашого розбиття, тобто перший потік отримає рядок від 0 – 25, другий від 25 – 50, третій 50-75, а четвертий від 75 – 100.

В випадках коли кількість рядків не кратна кількості потоків, наприклад 33 / 4 = 8.25, тоді всі потоки крім останнього отримають по 8 рядків, а останній потік забере у себе весь надлишок.

Вирізка з коду: for (int i = 0; i < num\_of\_threads; i++)

{

size\_t startRow = i \* step;

size\_t endRow = (i == num\_of\_threads - 1) ? n : startRow + step;

threads[i] = thread(minusMatricesParts, ref(A), ref(B), ref(C), startRow,endRow);

}

Тут чітко видно яким чином відбувається розбиття. Для першого потоку і = 0, його початковий рядок 0, а кінцевий 0 + step. Оскільки індексація в масиві іде з нуля, то 4 потік це і = 3, коли і == 3, тоді початковий рядок 3 \* крок, а кінцевий рядок це n, так ми запобігаємо

Випадкам коли к-сть рядків не кратна потокам.

Сама паралельність реалізована так, що задається кількість потоків, крок обраховується методом ділення кількості рядків на кількість потоків, створюється масив потоків і в середині циклу відбувається рівномірне розбиття елементів серед потоків, маємо додаткову функцію, яка додає лише задані інтервали матриці де і виконуються обчислення.

**Результати виконання:**

**Зображення, що містить текст, знімок екрана, Шрифт

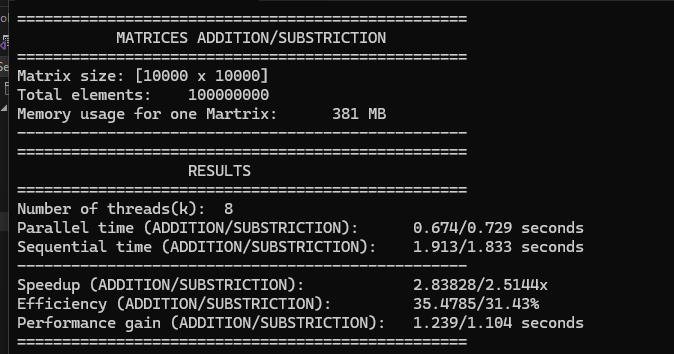
Автоматично згенерований опис**

**Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис**

**Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис**

****