ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Паралельні та розподілені обчислення

ЛАБОРАТОРНА РОБОТА №1

Тема: «Послідовне та паралельне додавання і віднімання матриць»

Виконав:

Ст. *Лук'янчук Денис*

Група ПМІ-33

2025

Тема: «Послідовне та паралельне додавання і віднімання матриць».

Мета роботи: Послідовно та паралельно виконати додавання/віднімання матриць, дослідити прискорення та ефективність паралельних обчислень.

Теоретичний матеріал

Послідовні обчислення – це спосіб виконання програми, при якому всі інструкції виконуються одна за одною в одному потоці управління. У випадку матричних операцій це означає, що кожен елемент матриці обробляється по черзі, незалежно від інших.

Переваги:

- Простота реалізації;
- Відсутність накладних витрат на синхронізацію;
- Передбачуваність результату.

Недоліки:

- Використовується лише одне ядро процесора, інші залишаються незадіяними;
- Час виконання зростає пропорційно до обсягу задачі.

Паралельні обчислення – це спосіб організації виконання програми, при якому завдання ділиться на підзадачі та виконується одночасно кількома потоками або процесами. У задачах обробки матриць це означає, що кожен потік може обчислювати окремі рядки чи блоки матриці.

Переваги:

- Використання кількох ядер процесора, що зменшує час обчислень;
- Підвищення продуктивності для великих задач;
- Можливість масштабування на багатопроцесорних системах.

Недоліки:

- Необхідність додаткової синхронізації між потоками;
- Накладні витрати на створення потоків;
- Для малих задач паралельний алгоритм може працювати повільніше, ніж послідовний.

Прискоренням називається відношення часу виконання послідовного алгоритму до часу виконання паралельного алгоритму на k потоках:

$$S = \frac{T_{seq}}{T_{par}}$$

де:

- T_seq– час виконання послідовного алгоритму;
- T_par– час виконання паралельного алгоритму.

Інтерпретація: якщо S = 3, це означає, що паралельний алгоритм виконується утричі швидше, ніж послідовний.

Ефективність показує, наскільки добре використані ресурси (потоки або ядра) при паралельному обчисленні:

$$E = \frac{S}{k} \cdot 100\%$$

де k – кількість потоків.

- Якщо E ≈ 100%, це означає, що ресурси використовуються максимально ефективно.
- Якщо E < 100%, це свідчить про втрати продуктивності, які можуть бути викликані накладними витратами на керування потоками, синхронізацію або нерівномірний розподіл навантаження.

Хід роботи

У цій лабораторній роботі була реалізована обробка матриць двома способами: **послідовним** і п**аралельним**. Програма дозволяє користувачу самостійно задавати **розмір матриці** (кількість рядків та стовпців) і **кількість потоків**, на які будуть розподілені обчислення.

1. **Послідовне обчислення:**
   - Для додавання та віднімання матриць кожен елемент нової матриці обчислюється по черзі за допомогою вкладених циклів.
   - Час виконання таких обчислень вимірювався за допомогою Stopwatch.

- Послідовне обчислення є базовим методом, на якому порівнюється ефективність паралельної реалізації.

2. **Паралельне обчислення:**
   - Матриця розбивається на частини, які одночасно обробляються кількома потоками (Task).
   - Кількість потоків визначається користувачем.
   - Паралельне обчислення дозволяє одночасно обробляти кілька рядків матриці, що значно зменшує загальний час обчислень на багатоядерних процесорах.
3. **Вирішення проблеми непарного розподілу рядків:**
   - У випадку, якщо **кількість рядків не ділиться на кількість потоків**, простий рівномірний розподіл рядків неможливий.
   - У моїй реалізації цей випадок враховано:
     1. Обчислюється базова кількість рядків на потік (rowsPerThread).
     2. Обчислюється залишок рядків (remainingRows), які не входять у рівномірний розподіл.
     3. Перші remainingRows потоків отримують **по одному додатковому рядку**, щоб всі рядки були оброблені.
   - Завдяки цьому рішення жоден рядок не залишиться необробленим, а навантаження між потоками розподілено максимально рівномірно.
4. **Оцінка продуктивності:**
   - Після обчислень для обох методів були зафіксовані **час виконання**.
   - Розраховано **прискорення** як співвідношення часу послідовного виконання до часу паралельного.
   - Розраховано **ефективність** як відсоткове співвідношення прискорення до кількості потоків.
   - Для великих матриць паралельні обчислення показали значне прискорення і високу ефективність.

```
Посл?довне додавання: 6,5108 ms
Паралельне додавання (16 поток?в): 2,4261 ms
Прискорення: 2,68
Ефективн?сть: 16,77%

Посл?довне в?дн?мання: 4,3849 ms
Паралельне в?дн?мання (16 поток?в): 2,5606 ms
Прискорення: 1,71
Ефективн?сть: 10,70%
```

(матриця 1000 на 1000, 16 потоків)

- Для малих матриць накладні витрати на створення потоків і синхронізацію можуть зменшувати ефективність паралельного виконання.

```
Посл?довне додавання: 0,3293 ms
Паралельне додавання (16 поток?в): 0,6398 ms
Прискорення: 0,51
Ефективн?сть: 3,22%

Посл?довне в?дн?мання: 0,2065 ms
Паралельне в?дн?мання (16 поток?в): 0,9429 ms
Прискорення: 0,22
Ефективн?сть: 1,37%
```

(матриця 100 на 100, 16 потоків)

5. **Висновки про оптимальне використання потоків:**

- Найкраща ефективність досягається, коли кількість потоків **приблизно відповідає кількості фізичних ядер процесора**.

- Якщо потоків більше, ядра змушені постійно перемикатися між ними, що створює додаткові витрати на планування та синхронізацію.

- Надлишок потоків може також погіршувати використання кеш-пам’яті, тому додаткові потоки не завжди забезпечують пропорційне прискорення.

Висновок: У ході лабораторної роботи було реалізовано послідовне та паралельне додавання і віднімання матриць із можливістю вибору розміру матриці та кількості потоків.

Паралельне виконання дозволяє значно скоротити час обчислень для великих матриць, проте для малих розмірів накладні витрати на створення потоків можуть зменшувати ефективність.

Оптимальна продуктивність досягається при кількості потоків, близькій до числа фізичних ядер процесора.