**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу  
Кафедра системного проектування**

**Звіт**

про виконання лабораторної роботи №1

«Дослідження базових операцій з потоками виконання»

Виконала:  
студентка IІІ курсу,

групи ДА-21  
Щербинко Єлизавета Миколаївна

Київ – 2024

**Мета роботи:** розглянути основні операції з потоками виконання, навчитися використовувати неблокуючу паралелізацію для вирішення найпростіших математичних задач, використовуючи обрану мову програмування. Навчитися досліджувати та оцінювати ефективність паралелізації алгоритму.

Завдання:

1. Визначити основні характеристики ПК, котрі на думку студента впливають на ефективність виконання паралельних обчислень. Зафіксувати значення даних характеристик для ПК студента, та занести їх до протоколу роботи.

2. Створити або використати наявній механізм для заміру часу виконання програми, або інших параметрів, котрі студент вважає релевантними. Занести опис механізму до роботи.

3. Вирішити обрану за варіантом задачу, не використовуючи паралелізацію. Заміряти час вирішення задачі, або інші параметри, котрі студент вважає релевантними.

4. Вирішити оборану за варіантом задачу, використовуючи паралелізацію. Заміряти час вирішення задачі, або інші параметри, котрі студент вважає релевантними. Обґрунтувати вибір алгоритму паралелізації (розподілення даних між потоками), надати опис та обґрунтування в протоколі роботи.

5. Перевірити алгоритм на фіксованих кількостях потоків: 2-рази меншій, ніж кількість фізичних ядер, на кількості рівній фізичним ядрам, на кількості рівній логічних ядрам, на кількості більшій в 2, 4, 8, 16 разів ніж кількість логічних ядер.

6. Повторити пункт 5 з використанням різної розмірності даних.

7. Зробити таблиці та графіки залежності часу виконання завдання від кількості потоків для різної розмірності даних. Надати опис графіків, з причинами виникнення отриманих результатів в протоколі роботи.

8. Надати висновок, що повинен містити аналіз отриманих результатів.

Варіант 23:



Хід роботи:

1. На мою думку, основними характеристиками, котрі впливають на ефективність виконання паралельних обчислень є кількість ядер і потоків, оперативна пам’ять, ssd та іноді відеокарта.

|  |  |
| --- | --- |
| Характеристики ноутбука | Значення |
| Ядра | 6 |
| Потоки, які обробляють ядра | 12 |
| RAM | 16 гб |
| SSD | 512 гб |

2. Замір часу реалізується як:

#include <ctime>

clock\_t startTime = clock();

for (int iter = 0; iter < *кількість повторень коду*; iter++){

// тут код, час виконання якого замірюється

}

clock\_t endTime = clock();

double seconds = (double (endTime - startTime)) / CLOCKS\_PER\_SEC;

cout << seconds << endl;

3. Вирішимо задачу без паралелізації (розмірність масиву = 10, кількість повторень програми = 10000).

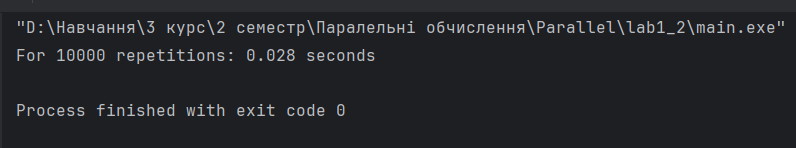


Рис. 1 – Послідовне рішення. Спроба 1

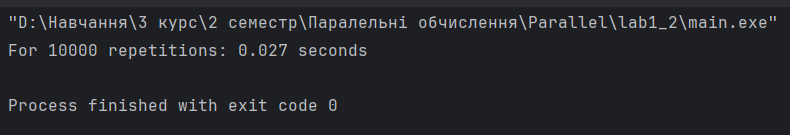


Рис. 2 – Послідовне рішення. Спроба 2

4. Вирішимо задачу з паралелізацією (розмірність масиву = 10, кількість повторень програми = 10000, кількість потоків = 6).

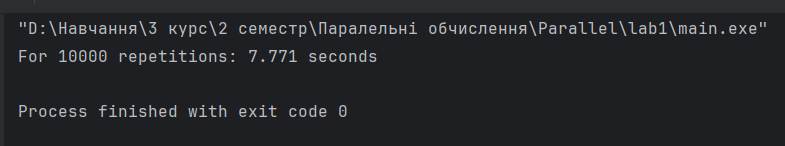


Рис. 3 – Паралельне рішення. Спроба 1

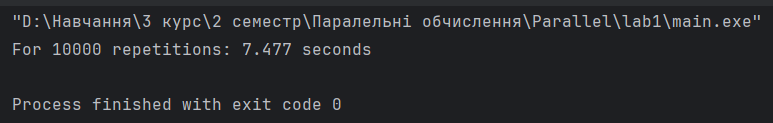


Рис. 4 – Паралельне вирішення. Спроба 2

Кількість потоків збільшимо вдвічі:

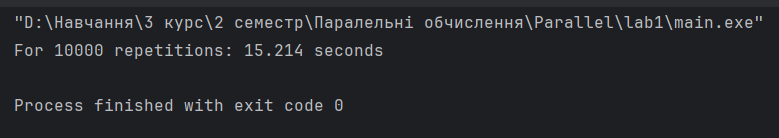


Рис. 5 – Паралельне рішення. Спроба 3

Кількість потоків зменшимо вдвічі:

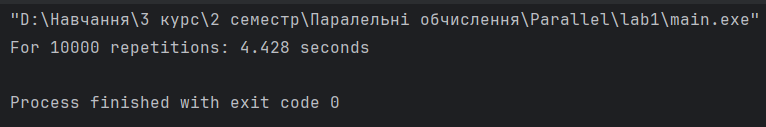


Рис. 6 – Паралельне рішення. Спроба 4

У коді, де використано паралельну обробку, уся матриця розбивається на threads частин, кожна з яких відповідає певному набору рядків, кожен потік отримує свою частину матриці та заповнює її незалежно. Аналогічно до заповнення, для дзеркального відображення вся матриця розбивається на частини, де кожен потік обробляє певний діапазон рядків і відображає числа в додаткову матрицю, щоб уникнути конфліктів запису.

5. Перевіримо алгоритм на інших параметрів. Спочатку порівняємо послідовний та паралельний (кількість потоків = 6). Кількість повторень програми = 1000.

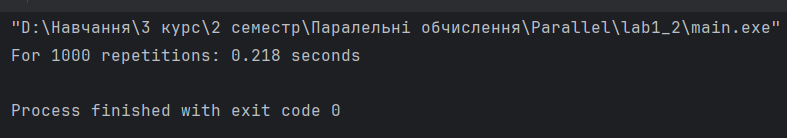


Рис. 7 – Послідовний алгоритм. Розмірність матриці = 100.

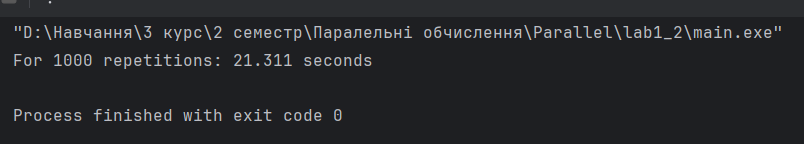


Рис. 8 – Послідовний алгоритм. Розмірність матриці = 1000.

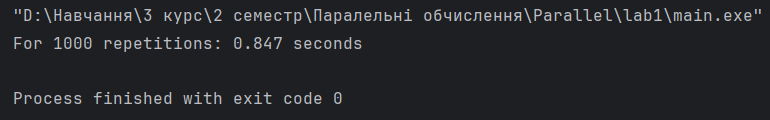


Рис. 9 – Паралельний алгоритм. Розмірність матриці = 100.

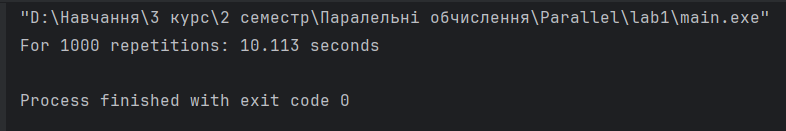


Рис. 10 – Паралельний алгоритм. Розмірність матриці = 1000.

Тепер порівняємо паралельний алгоритм з розмірністю матриці = 1000, з різною кількістю потоків.

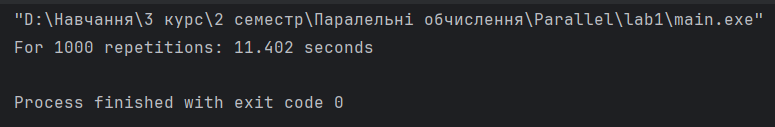


Рис. 11 – 3 потоки

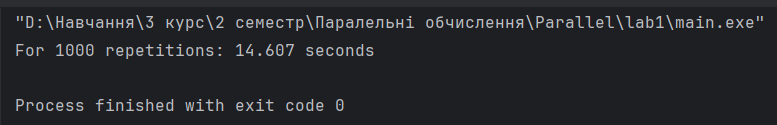


Рис. 12 – 2 потоки

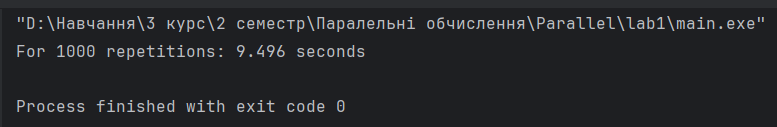


Рис. 13 – 12 потоків

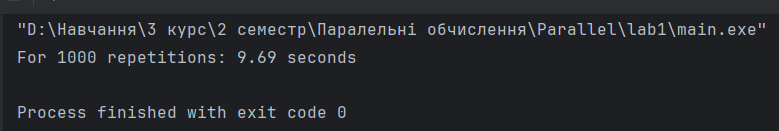


Рис. 14 – 24 потоки

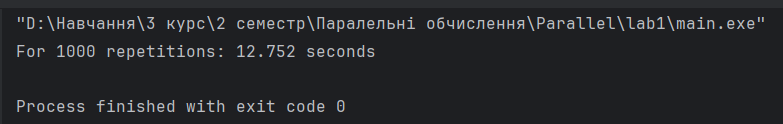


Рис. 15 – 48 потоків

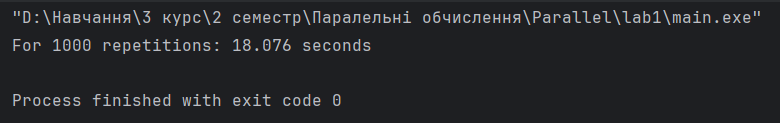


Рис. 16 – 96 потоків

6. Повторимо таке саме на розмірності матриці 10000. Через велику розмірність кількість повторень зменшимо до 10.

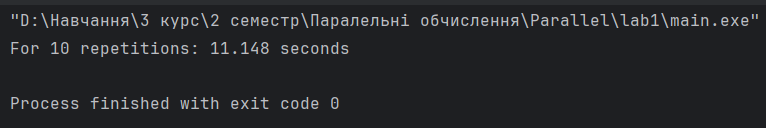


Рис. 17 – 6 потоків

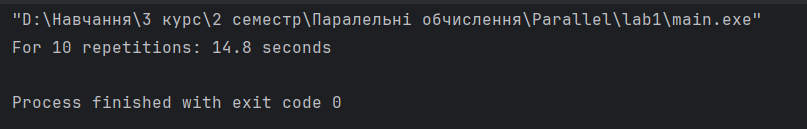


Рис. 18 – 3 потоки

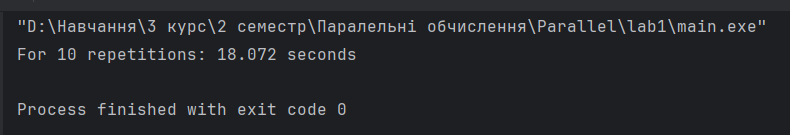


Рис. 19 – 2 потоки

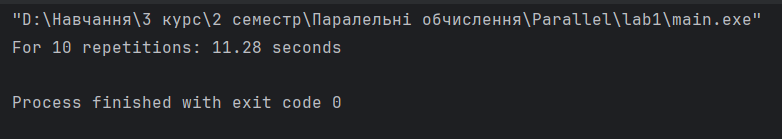


Рис. 20 – 12 потоків

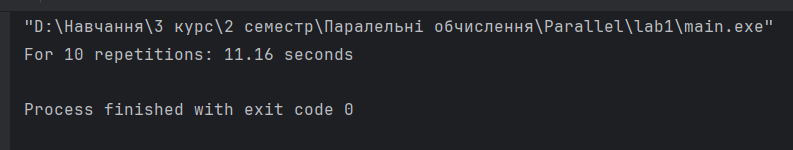


Рис. 21 – 24 потоки

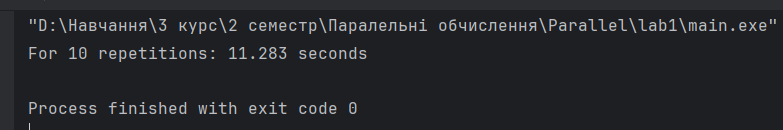


Рис. 22 – 48 потоків

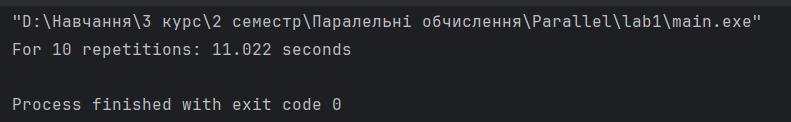


Рис. 23 – 96 потоків

Повторимо на розмірності 100 з 1000 повторів:

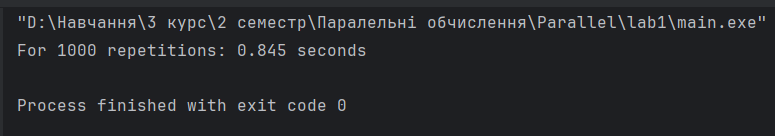


Рис. 24 – 6 потоків

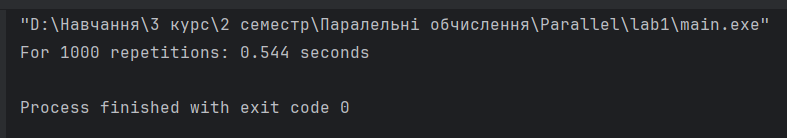


Рис. 25 – 3 потоки

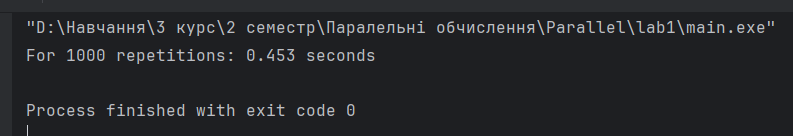


Рис. 26 – 2 потоки

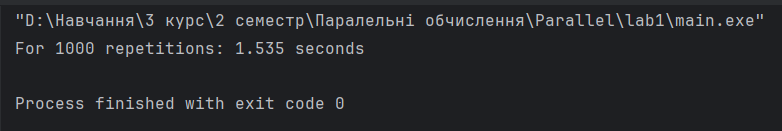


Рис. 27 – 12 потоків

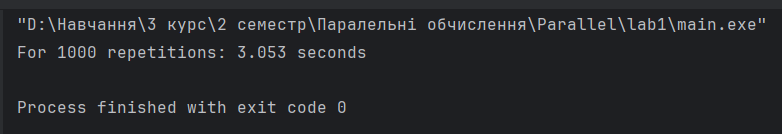


Рис. 28 – 24 потоки

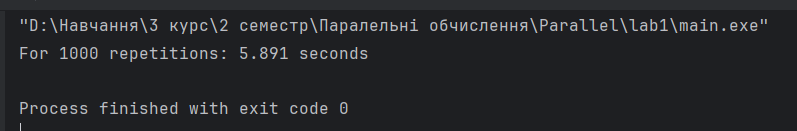


Рис. 29 – 48 потоків

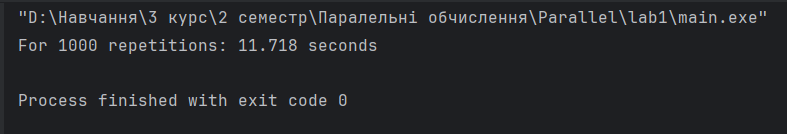


Рис. 30 – 96 потоків

7. Намалюємо графіки:



Рис. 31 – Графік 1



Рис. 32 – Графік 2



Рис. 33 – Графік 3

Як показано на графіках, при невеликому обсязі даних (матриця розміром 100\*100) послідовне виконання програми є більш оптимальним і доцільним (на рис. 7 та 26 продемонстровано, що послідовний підхід швидший вдвічі) . Однак зі збільшенням обсягу даних використання потоків стає більш актуальним. Чим більші дані, тим більшою стає потреба у паралельній обробці.

**ВИСНОВОК**

Під час лабораторної роботи було розроблено дві програми для віддзеркалення матриці відносно побічної діагоналі, використовуючи різні підходи: послідовний та паралельний.

У паралельній реалізації вся матриця розбивається на threads частин, кожна з яких відповідає певному набору рядків. Кожен потік отримує свою частину матриці та обробляє її незалежно. Аналогічно, під час віддзеркалення матриця розподіляється на частини, де кожен потік обробляє певний діапазон рядків і записує відображені значення в додаткову матрицю, щоб уникнути конфліктів запису.

У ході дослідження було визначено, що послідовний підхід є більш доцільним для невеликих обсягів даних. Натомість паралельна обробка дозволяє оптимізувати виконання програми при роботі з більшими матрицями, що підтверджують графіки 2 і 3 (рис. 32 і 33).