КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Звіт

з лабораторної роботи

“Розподілене та паралельне програмування”

з теми

“Розв’язок задачі Two Sum за допомогою послідовних та паралельних

обчислень”

Виконав

Студент 4 курсу

Групи ТК-41

Шевчук Кіріл Юрійович

Київ – 2024

Contents

[Постановка задачі 3](#_Toc164203529)

[Алгоритм 3](#_Toc164203530)

[Дані 3](#_Toc164203531)

[Результати виконання 4](#_Toc164203532)

[Висновок 5](#_Toc164203533)

# Постановка задачі

Використовуючи OpenMP і MPI реалізувати послідовний на паралельні алгоритми знаходження розв’язку задачі 2 Sum, порівняти час виокнання і визначити як використання паралельних алгоритмів прискорює знаходження результату.

Задача Sum 2 – дано масив чисел і цільове значення, необхідно знайти пару чисел таку, що їх сума дорівнює цільовому значенню.

# Алгоритм

Спочатку ми фіксуємо один з елементів масиву під назвою x\_i. Далі обчислюємо різницю між загальною сумою (total) та значенням цього фіксованого елементу. Після цього перевіряємо, чи існує ця різниця як елемент у масиві шляхом перебору всіх елементів.

У паралельних варіантах алгоритму кожен процес обробляє тільки частину масиву. Якщо є N процесів, кожен процес під номером P буде аналізувати підмасив, який починається з індексу (count \* P / N) та закінчується на індексі (count \* (P+1) / N), де count - кількість елементів у масиві. Це дозволяє паралельно виконувати пошук потрібного числа, ефективно розподіляючи обчислення між процесами.

# Дані

Цільове значення: 600

Масив: масив із N чисел

Процесор: Ryzen 7600x

Кількість ядер: 6

Кількість логічних процесорів: 12

Швидкість: 4.7 GHz – 5.3 GHz

# Результати виконання

6 паралельних процеси

Вплив розміру масиву на час виконання

|  |  |  |  |
| --- | --- | --- | --- |
| N | Sequential | OpenMP | MPI |
| 1\_000 | 0.0008 | 0.0006 | 0.0002 |
| 10\_000 | 0.0614 | 0.0150 | 0.0245 |
| 100\_000 | 6.2249 | 1.4112 | 2.1194 |
| 500\_000 | 155.2757 | 33.7326 | 34.3565 |

Вплив кількості процесів на час виконання (500\_000 елементів)

|  |  |  |  |
| --- | --- | --- | --- |
| Proc\_num | Sequential | OpenMP | MPI |
| 1 | 155.2757 | 156.3275 | 157.3773 |
| 3 | - | 58.1105 | 64.01111 |
| 4 | - | 45.8033 | 56.8580 |
| 6 | - | 33.7326 | 34.3565 |
| 12 | - | 26.7232 | 28.7543 |

# Висновок

За результатами виконання видно, що зі збільшенням кількості вхідних даних квадратично збільшується і час виконання, незалежно від кількості процесів.

При цьому, збільшення кількості процесів веде до зниження часу виконання, що підкреслює переваги паралельних обчислень для оптимізації роботи з великими даними. Оскільки складність алгоритму завжди O(N^2), при розділенні на процеси, кожен процес просто окремо виконує задачу із N^2 / M ітерацій.

Серед двох використаних технологій, OpenMP показав трохи кращі результати за швидкістю обробки.