КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП’ЮТЕРНИЙ НАУК ТА КІБЕРНЕТИКИ

Звіт

до лабораторної роботи з курсу:  
«Розподілене та паралельне програмування»

Виконавстудент 4 курсу, групи ТТП-41, Спотар Владислав Володимирович

Київ 2023

**OpenMP**

**OpenMP** – АРІ для написання багатопоточних програм з спільною

пам’яттю, який включає набір директив компілятору та бібліотечних функцій.

Тобто, це стандартизований та документований інструмент, що значно

спрощує процес створення багатопоточних програм на Fortran, C та С++,

розділяючи роботу програми на менші частини, які потім можуть

виконуватися одночасно на декількох процесорах або ядрах процесора.

**OpenMP** дозволяє розпаралелювати цикли за допомогою директиви *#pragma*

*omp parallel for*.

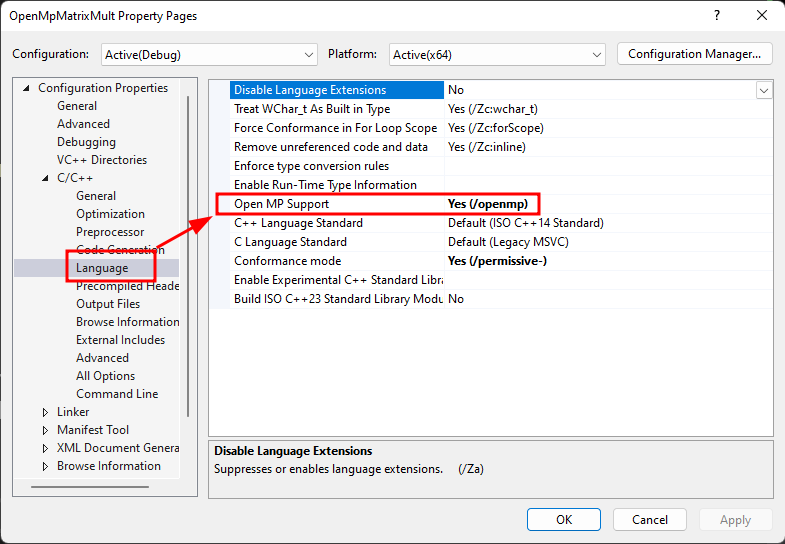
Директива *#pragma omp sections* дозволяє створювати секції коду, які

виконуються паралельно.

OpenMP має механізми для синхронізації потоків, такі як критичні секції

*(#pragma omp critical)* та атомарні операції *(#pragma omp atomic).*

Для запуску програми до проекту було підключено підтримку openMP



**Постановка задачі.**Реалізувати алгоритм множення матриць.

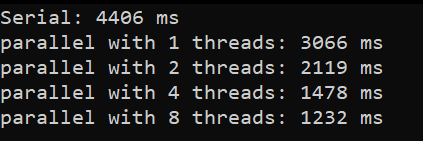
За допомогою технології OpenMP реалізувати паралельний алгоритм.

Порівняти ефективність виконання.

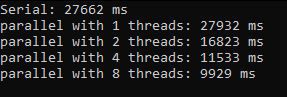
**Тестування:**

Для тестування запускаємо код на різній кількості даних, а також для різної кількості ядер.

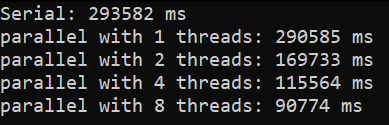
1. 500\*500:



1. 1000\*1000:



1. 2000\*2000



**MPI**

MPI (Message Passing Interface) - це стандарт для паралельного програмування,

який дозволяє обмінюватися даними та взаємодіяти між різними процесами або вузлами обчислювального кластера. Він часто використовується у

високопродуктивних обчисленнях для створення програм, що виконуються на

кластерах комп'ютерів або в паралельному середовищі.

Основні поняття MPI включають точки входу, групи та комунікатори. Кожна

програма, яка використовує MPI, складається з різних процесів, які

обмінюються повідомленнями для виконання різних завдань. Кожен процес

може мати свій власний ранг та комунікатор, які використовуються для

ідентифікації та взаємодії з іншими процесами.

Деякі основні функції MPI включають MPI\_Init для ініціалізації середовища

MPI, MPI\_Comm\_rank для визначення рангу процесу, MPI\_Comm\_size для

визначення загального числа процесів та MPI\_Send/MPI\_Recv для надсилання

та отримання повідомлень між процесами. Також є багато інших директив та

функцій, які дозволяють ефективно взаємодіяти між процесами та керувати їх

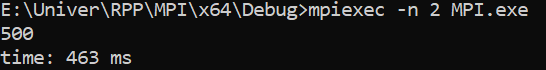
виконанням.

**Тестування:**

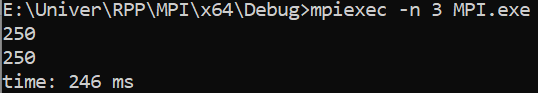
Для тестування запускаємо код на різній кількості даних, а також для різної кількості ядер.

1. **500\*500:**

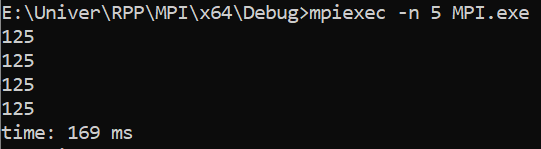
* 1 ядро:



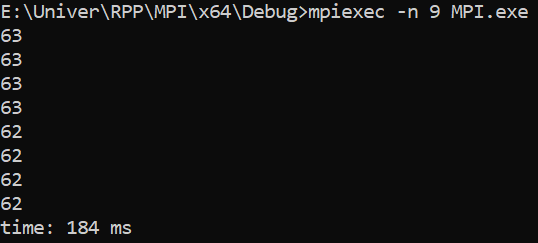
* 2 ядра:



* 4 ядра:

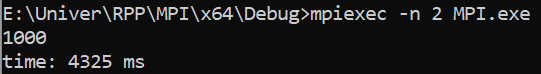


* 8 ядер:

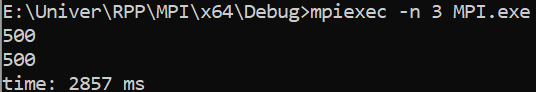


1. **1000\*1000:**

* 1 ядро:



* 2 ядра:



* 4 ядра:



* 8 ядер:



1. **1000\*1000:**

* 1 ядро:



* 2 ядра:



* 4 ядра:



* 8 ядер:



**Висновок**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Розмір матриці | Послідовне | К-сть ядер | OpenMp | MPI |
| 500x500 | 4 406 ms | 1 | 3 066 ms | 463 ms |
| 2 | 2 119 ms | 246 ms |
| 4 | 1 478 ms | 169 ms |
| 8 | 1 232 ms | 183 ms |
| 1000x1000 | 26 662 ms | 1 | 27 932 ms | 4 325 ms |
| 2 | 16 823 ms | 2 857 ms |
| 4 | 11 533 ms | 2 269 ms |
| 8 | 9 929 ms | 2 528 ms |
| 2000x2000 | 293 582 ms | 1 | 290 582 ms | 68 869 ms |
| 2 | 169 733 ms | 39 361 ms |
| 4 | 115 564 ms | 32 620 ms |
| 8 | 90 774 ms | 30 931 ms |

**OpenMP:**

Як можна побачити з таблиці порівняння, по-перше, послідовне виконання по часу на великій кількості даних співпадає з часом виконання паралельного з 1 ядром. Це говорить про те, що якщо на машині є тільки одне ядро – то не треба перероблювати код на послідовний, OpenMP сама про це подбає і не буде брати в розрахунок директиви розпаралелення.

В усіх трьох випадках зі збільшенням кількості ядер для OpenMP зменшується час виконнаня. Найпомітніше це видно з переходу з 1 ядра на 2, де прирост відбувається в 1,5-2 рази. Далі при збільшенні ядер час виконання зменшується, але не пропорційно збільшенню ядер.

**MPI:**

Чітко видно, що MPI працює набагато ефективніше виконуючу ту саму задачу. Зі збільшенням кількості даних різниця в швидкості виконання завжди тримається так, що MPI в 6 разів швидше при одному воркері. При збільшенні кількості ядер ця різниця зменшується.

Отже, використання OpenMP та MPI прискорює виконання алгоритму (за умови використання більше одного ядра). Однак, при збільшенні кількості ядер використання MPI дає суттєвіший результат в ефективності в порівнянні з OpenMP.