Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки

*Кафедра інформатики та програмної інженерії*

Звіт до лабораторної роботи №7

з курсу

“Технології паралельних обчислень”

# **Виконав:**

Студент гр. ІТ-02

Терешкович М.О.

Київ – 2023

**Тема:** Розробка паралельного алгоритму множення матриць з використанням МРІ-методів колективного обміну повідомленнями («один-до-багатьох», «багато-до-одного», «багато-до-багатьох») та дослідження його ефективності

**Завдання:**

Ознайомитись з методами колективного обміну повідомленнями типу «один-до-багатьох», «багато-до-одного», «багато-до-багатьох» (див. лекцію та документацію стандарту MPI).

Реалізувати алгоритм паралельного множення матриць з використанням розподілених обчислень в MPI з використанням методів колективного обміну повідомленнями. **40 балів**.

Дослідити ефективність розподіленого обчислення алгоритму множення матриць при збільшенні розміру матриць та при збільшенні кількості вузлів, на яких здійснюється запуск програми. Порівняйте ефективність алгоритму при використанні методів обміну повідомленнями «один-до-одного», «один-до-багатьох», «багато-до-одного», «багато-до-багатьох». **60 балів**.

**Хід роботи:**

**Додаткові методи та класи:**

Метод Print() для виводу матриці в консоль:

**A screen shot of a computer code

Description automatically generated with low confidence**

A picture containing text, screenshot, font

Description automatically generatedМетод для заповнення масиво числами:

Методи перетворення між двовимірними та одновимірними масивами:

A picture containing text, screenshot, font

Description automatically generated

Схожі методи вже були розглянуті в інших лабараторних, тому не бачу сенсу ще раз проходитися по-рядково по ним.

**Множення матриць з колективними методами обміну повідомленнями один-до-багатьох:**

**A picture containing text, font, screenshot

Description automatically generated**

Для початку програма визначає константні змінні SIZE та MASTER. SIZE представляє розмір матриць, а MASTER - ранг головного процесу (процес з рангом 0). Усередині методу main відбувається ініціалізація MPI за допомогою MPI.Init(args).

Далі поточний ранг процесу та загальна кількість процесів отримуються за допомогою функцій MPI.COMM\_WORLD.Rank() та MPI.COMM\_WORLD.Size(). Після чого код перевіряє, чи кількість рядків у матрицях (SIZE) ділиться на кількість процесів (processesCount). Якщо не ділиться, програма виводить повідомлення про неможливість розподілу для заданої кількості процесорів. Потім MPI завершується з допомогою MPI.Finalize(), і програма повертається.

Якщо ж кількість рядків ділиться на кількість процесів, то змінна rowsPerProcess обчислюється як кількість рядків на процес (SIZE / processesCount). Далі масиви та змінні оголошуються для зберігання та виконання множення матриць. matrixA, matrixB та matrixC - двовимірні масиви, що представляють матриці A, B та C відповідно. sentMatrixA1D, recvMatrixA1D, matrixB1D та matrixC1D - одновимірні масиви, що використовуються для надсилання та отримання даних між процесами. subMatrixA та resultMatrix - підматриці, що використовуються для локальних обчислень.

A screen shot of a computer program

Description automatically generated with low confidence

Якщо поточний процес є головним (ранг 0), матриця A та матриця B заповнюються значеннями методом Array.arrayFill (в даному випадку матриця заповнюється всюди 1), а потім перетворюються в 1D масиви методом Array.convert2DTo1D.

Одномірний масив sentMatrixA1D розсіюється між усіма процесами за допомогою MPI.COMM\_WORLD.Scatter. Кожен процес отримує частину даних (SIZE \* rowsPerProcess) у масив recvMatrixA1D.

1D масив matrixB1D транслюється всім процесам за допомогою MPI.COMM\_WORLD.Bcast. Усі процеси отримують однакові дані у своїх масивах matrixB1D.

Масив recvMatrixA1D перетворюється назад у двовимірну підматрицю (subMatrixA) за допомогою методу Array.convert1DTo2D. Аналогічно масив matrixB1D перетворюється у двовимірну матрицю (matrixB). Після чого кожен процес виконує локальне множення матриць між subMatrixA та matrixB, зберігаючи результат у двовимірному масиві resultMatrix.

Масив resultMatrix знову перетворюється в одномірний масив за допомогою Array.convert2DTo1D, і всі процеси надсилають свої локальні результати до головного процесу за допомогою MPI.COMM\_WORLD.Gather.

A screen shot of a computer screen

Description automatically generated with low confidence

Під кінець, якщо поточний процес є головним, отримана матриця C1D перетворюється назад у двовимірну матрицю (матрицю C) за допомогою Array.convert1DTo2D. Отримана матриця C друкується за допомогою Array.printMatrix і виводиться час виконання.

Нарешті, MPI завершується за допомогою MPI.Finalize().

A picture containing text, font, screenshot

Description automatically generated

Це був метод обміну повідомленнями типу «один-до-багатьох» з використанням MPI.COMM\_WORLD.Gather(). Ця функція збирає дані з усіх процесів до кореневого процесу.

**Множення матриць з колективними методами обміну повідомленнями багато-до-багатьох:**

У цьому варіанті роботи весь код той самий але тепер замість методу Gather ми будемо використовувати Allgather, який збирає дані з усіх процесів і розподіляє їх між усіма процесами.



Як результат спочатку ми використали MPI.COMM\_WORLD.Scatter() для розподілу даних матриці A (sentMatrixA1D) від кореневого процесу (MASTER) до кожного процесу, де кожен процес отримує свою частину даних у recvMatrixA1D. Потім ми застосували MPI.COMM\_WORLD.Bcast()

для трансляції даних матриці B (матрицяB1D) від кореневого процесу (MASTER) до всіх інших процесів, де кожен процес отримує дані у матрицюB1D. Після чого, в залежності від завдання ми використали або MPI.COMM\_WORLD.Allgather() або MPI.COMM\_WORLD.Gather().

**Приклади роботи програми:**

Для початку створимо дві матриці 12\*12 яка буде заповнена всюди 1 для перевірки коректності роботи множення:

Як бачимо все працює правильно, в результаті множеня матриці А розмірністю 12\*12 і матриці В розмірності 12\*12 ми отримали матрицю С.

A screenshot of a computer screen

Description automatically generated with medium confidence

Тепер запустимо для різної розмірності і з різною кількістю процесорів:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**Дослідження:**

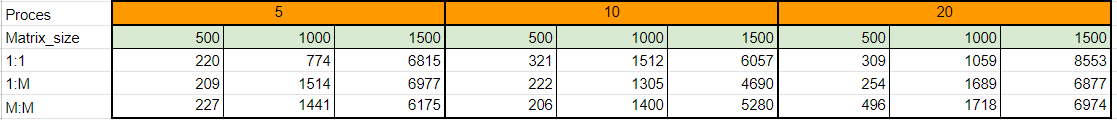


Рисунок 1 – Тестування швидкості множення матриць з методами обміну повідомленнями «1:1», «1:M» та «M:M»

**Висновок:**

В результаті виконання лабораторної роботи було розроблено паралельний алгоритм множення матриць з використанням МРІ-методів колективного обміну повідомленнями. Для реалізації алгоритму були використані методи обміну повідомленнями типу "один-до-багатьох", "багато-до-одного" та "багато-до-багатьох", засновані на стандарті MPI.

У процесі дослідження ефективності розподіленого обчислення алгоритму множення матриць були проведені експерименти зі збільшенням розміру матриць і кількості вузлів, на яких запускалася програма. Також порівнювалася ефективність алгоритму при використанні різних методів обміну повідомленнями, зокрема "один-до-одного", "один-до-багатьох", "багато-до-одного" та "багато-до-багатьох".

Результати експериментів показали, що паралельний алгоритм множення матриць з використанням МРІ-методів колективного обміну повідомленнями є ефективним засобом за неколективні методи. При збільшенні розміру матриць та кількості вузлів спостерігалося покращення швидкості виконання алгоритму. Враховуючи різні методи обміну повідомленнями, виявлено, що метод "один-до-багатьох" найбільш ефективний для вирішення даної задачі.