Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Комп’ютерний практикум №7**

з дисципліни «Технології паралельних обчислень»

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»»

|  |  |
| --- | --- |
| **Виконав:**  студент групи ІТ-04  Коновальчук Андрій Володимирович  Дата здачі \_\_\_\_\_\_\_\_  Захищено з балом \_\_\_\_\_\_\_ | **Перевірила**:  асистент кафедри ІПІ  Дифучина Олександра Юріївна |

Київ 2023

**Хід роботи:**

***Завдання 1:*** Ознайомитись з методами колективного обміну повідомленнями типу «один-до-багатьох», «багато-до-одного», «багато-до-багатьох».

*Опис пункту:*

Для виконання подальших пунктів ознайомились з методами MPI.COMM\_WORLD.Scatterv, MPI.COMM\_WORLD.Bcast, MPI.COMM\_WORLD.Barrier та MPI.COMM\_WORLD.Gatherv.

***Завдання 2:*** Реалізувати алгоритм паралельного множення матриць з використанням розподілених обчислень в MPI з використанням методів колективного обміну повідомленнями.

*Опис пункту та фрагменту коду:*

Задача будується наступним чином:

1. Ініціалізуємо комунікатор, отримуємо розмір комунікатора та індекс поточного процесу. Розбиваємо матрицю на рівні частини та зберігаємо остачу.
2. Створюємо та заповнюємо масиви, що містять:
   1. (sendcount) кількість елементів, які передаються з процесу-відправника до кожного процесу-одержувача.
   2. (displs) зміщення вхідних і вихідних даних для кожного процесу в масиві даних.
   3. (recvcount) кількість елементів, які приймає кожен процес в результаті операції Scatterv, котрий має однакове значення з sendcount для поточного процесу.
3. Створюємо підматрицю розміром (recvcount) х (SIZE). Та викликаємо метод Scatterv, яка розподіляє частини матриці A між усіма процесами в комунікаторі.
4. Викликаємо метод Bcast, котра транслює матрицю B на всі процеси в комунікаторі.
5. Викликаємо метод Barrier для блокування виконання всіх процесів в комунікаторі до тих пір, поки всі вони не досягнуть цієї точки у своєму виконанні. Метод використовується для забезпечення синхронізації всіх процесів перед виведенням інформації про те, які процеси виконують який кусок роботи (printf, “start, end, worker”).
6. Кожен процес обчислює свою частину матричного добутку submatrixA та відправляє її назад на головний процес за допомогою функції MPI.COMM\_WORLD.Gatherv(). На головному процесі отримані частини матричного добутку збираються в кінцеву матрицю resultMatrix.
7. Загальний час роботи алгоритму вимірюється за допомогою функції System.nanoTime() та виводиться на екран. Також можна відобразити результуючу матрицю, якщо drawFlag має значення true.

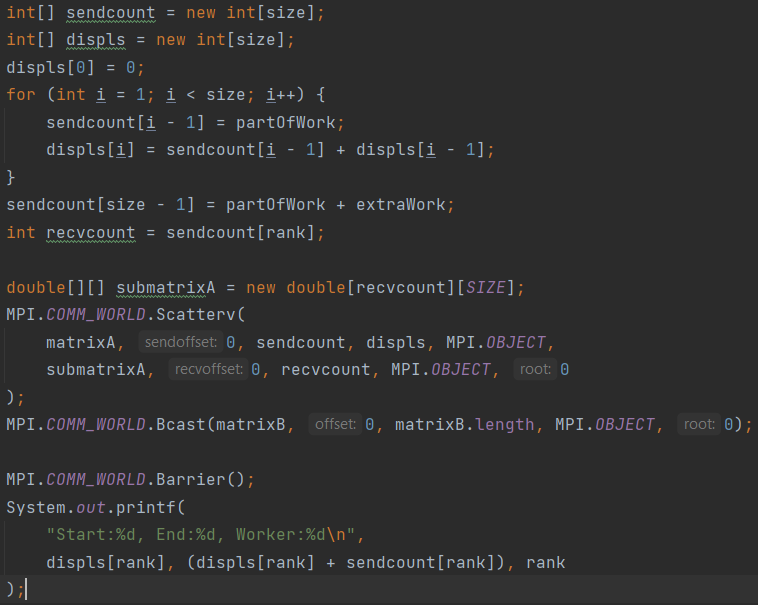


Рисунок 1 – Розподіл матриці A між процесами, трансляція матриці B, вивід інформації про роботу процесів

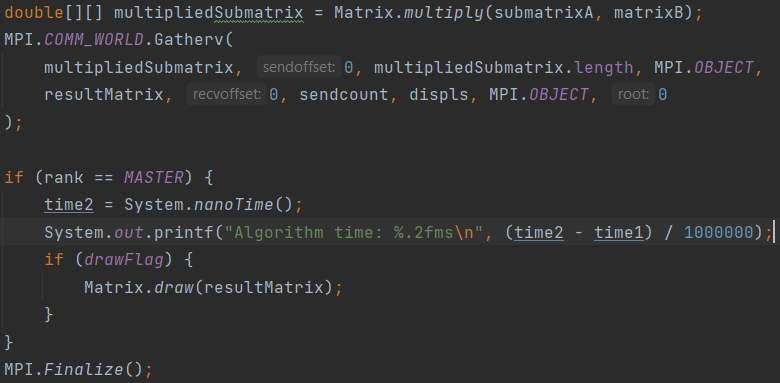


Рисунок 2 – Множення підматриці A і матриці B процесами, відправка результату головному процесу, збір у результуючу підматрицю, швидкодія

*Спостереження, результати, висновки:*

Запускаємо код з величиною матриці 100 та кількістю робочих вузлів 6.

В консолі бачимо частини роботи розділені між 6-ма воркерами де worker-5 отримав зверху решту роботи. Також бачимо швидкість обчислення у мілісекундах та виведену матрицю (множення правильне).

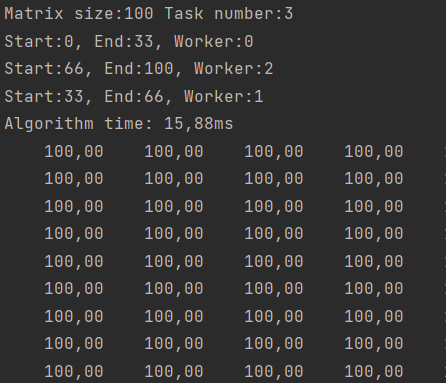


Рисунок 3 – Результати роботи алгоритму

***Завдання 3:*** 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».

*Опис пункту, спостереження, результати, висновки:*

Для виконання завдання, ми будемо змінювати розмір комунікатора, змінюючи значення параметрів запуску від 3 до 12, і будемо варіювати розмір матриці від 500 до 2500. Запускати будемо код цієї та попередньої лабораторної. Адже нас цікавлять результати роботи блокуючого та колективних методів.

Таблиця 1 – Колективні методи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TIME, ms | Workers number | | | |
| Size | 3 | 6 | 9 | 12 |
| 500 | 128,64 | 140,16 | 146,74 | 174,32 |
| 1000 | 1373,03 | 1087,31 | 1337,17 | 1410,21 |
| 1500 | 8315,91 | 6224,97 | 6569,09 | 6540,40 |
| 2000 | 25213,32 | 16047,04 | 15777,92 | 15774,81 |
| 2500 | 62022,72 | 36426,75 | 36678,72 | 35806,50 |

Таблиця 2 – Блокуючі методи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TIME, ms | Workers number | | | |
| Size | 3 | 6 | 9 | 12 |
| 500 | 155,67 | 158,71 | 175,48 | 158,43 |
| 1000 | 1671,26 | 1068,31 | 1298,02 | 1436,44 |
| 1500 | 12610,82 | 5965,63 | 6032,59 | 6475,70 |
| 2000 | 38197,55 | 17492,19 | 16279,03 | 17032,51 |
| 2500 | 87917,73 | 38788,58 | 36855,29 | 36432,59 |

Рисунок 4 – Колективні методи

Рисунок 5 – Блокуючі методи

Рисунок 6 – Worker - 3

Рисунок 7 – Worker - 6

Рисунок 8 – Worker - 9

Рисунок 9 – Worker - 12

Рисунок 10 – Size - 500

Рисунок 11 – Size - 1000

Рисунок 12 – Size - 2000

Рисунок 13 – Size - 2500

Висновок:

Як і очікувалось, в цілому, паралельні обчислення з колективними методами обміну повідомленнями виконуються швидше ніж з блокуючими. Різниця більш відчутна на більших розмірах матриць (26 секунд на 3 воркерах, 2,4 секунди на 6-ти).

Збільшення кількості вузлів впливає по різному на різних величинах матриці. Так бачимо, що на графіках з розмірами матриць 500, 1000, 2000 та 2500, графіки мають різні форми, і досягають максимальної ефективності на різній кількості вузлів (відповіно 3, 6, 9, 12). Отже, для більшої швидкості обчислення необхідно правильно підбирати кількість вузлів.

**Висновок:**

В ході виконання практикуму було побудовано алгоритм паралельного обчислення матриць з використанням МРІ-методів обміну повідомленнями “один-до-багатьох”, “багато-до-одного” з використанням колективних методів. Було проведено експерименти, зібрано статистику результатів роботи алгоритмів за різних вхідних даних та робочих вузлів, зроблено висновки та порівняння роботи алгоритму.

У звіті наявні скріншоти з прикладами результатів роботи коду та їх фрагментів, а також опис роботи коду та пояснення результатів, графіки та висновки до швидкодії алгоритмів.

**Лістинг коду у вигляді GitHub-репозиторію:**

<https://github.com/m4cy43/parallel_programming/tree/master/lab7>