Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Комп’ютерний практикум №8**

з дисципліни «Технології паралельних обчислень»

Тема: «Розробка алгоритмів для розподілених систем клієнт-серверної архітектури»

|  |  |
| --- | --- |
| **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:*** Розробити веб-застосування клієнт-серверної архітектури, що реалізує алгоритм множення матриць або інший, який був Вами реалізований в рамках курсу «Технології паралельних обчислень», на стороні сервера з використанням паралельних обчислень. Розгляньте два варіанти реалізації 1) дані для обчислень знаходяться на сервері та 2) дані для обчислень знаходяться на клієнтській частині застосування.

*Опис пункту:*

Застосунок складається з 5 файлів (класів): Server, Client, Multiplication, MultiplicationThread, Generation. Server відповідає за серверну частину, Client за клієнтську, Multiplication відповідає за поділ матриці на частини між потоками, MultiplicationThread за перемноження частин в потоках, Generation за генерацію матриць двома способами – фіксоване число вздовж головної діагоналі та генерація однакових рядків по зростанню.

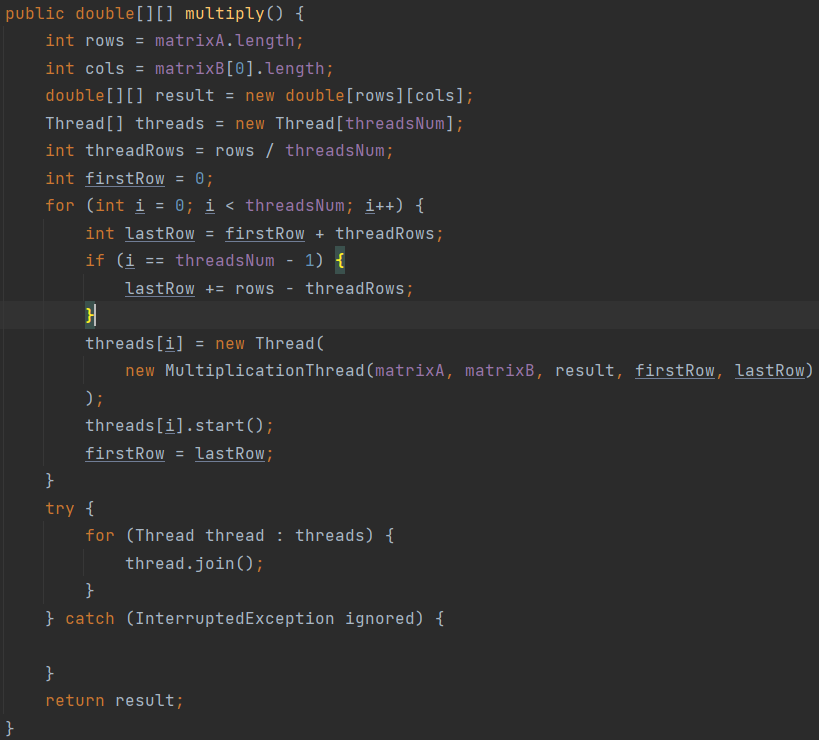


Рисунок 1 – Клас Multiplication

На рисунку 1 бачимо клас Multiplication та метод multiply. Метод ділить між кожним потоком рівну кількість колонок матриці та додає остачу до останньої колонки. Далі в метод run класу MultiplicationThread пересилається індекси колонок та самі матриці, результуюча матриця, де відбувається їх перемноження і запис результату у результуючу матрицю.

Клас Client (рисунок 2) ініціалізує сокет за спільним з сервером портом, буфер, потоки вводу та виводу. Користувач вводить в консоль опцію (данні на сервері чи на клієнті) і в залежності від вибору відбувається генерація матриць та пересилання даних на сервер (розмір матриці, кількість потоків, команда). Починається запис часу для обчислення швидкодії та зчитується результуюча матриця.

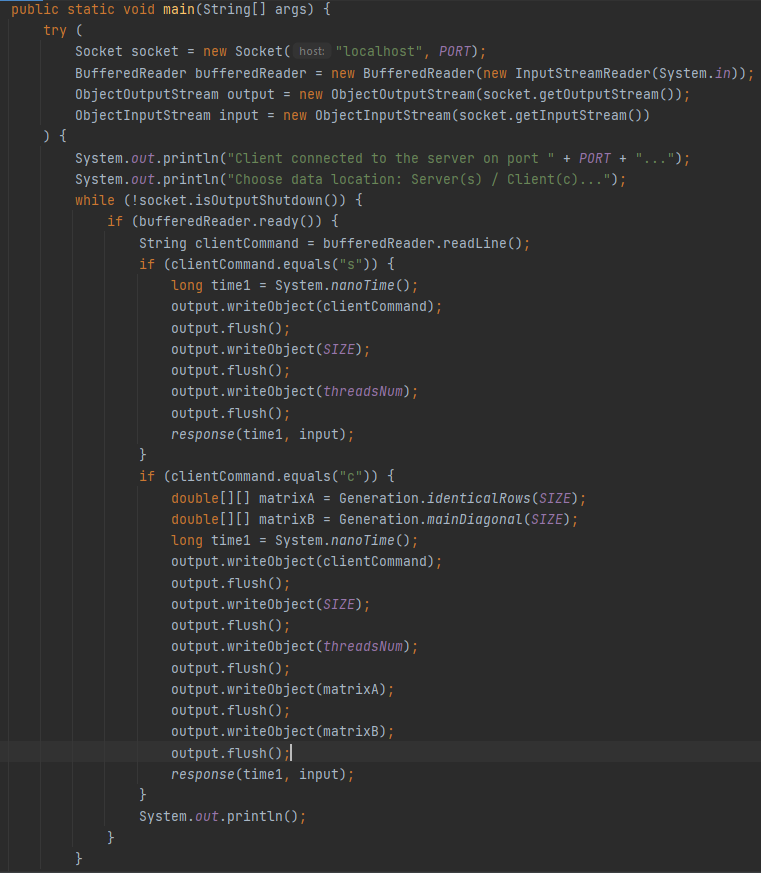


Рисунок 2 – Клас Client

На сервері (рисунок 3) ініціалізується сервер-сокет зі спільним портом із клієнтом та потоки вводу та виводу. Далі зчитуються данні з клієнта та в залежності від опції відбувається генерація матриць та виконання методі multiply класу Multiplication. В кінці на клієнт пересилається результуюча матриця. А на самому клієнті закінчується запис часу швидкодії алгоритму.

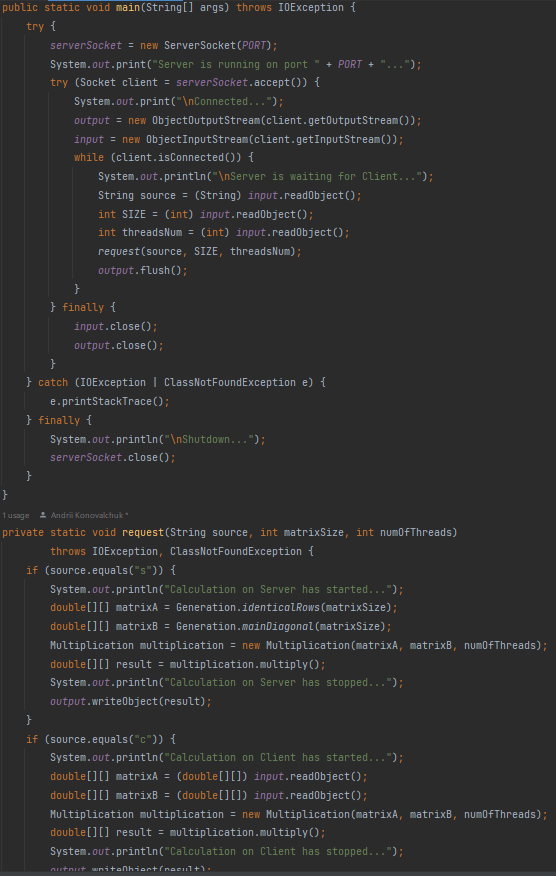


Рисунок 3 – Клас Server

Результати роботи алгоритму – вивід в консоль (рисунок 4).

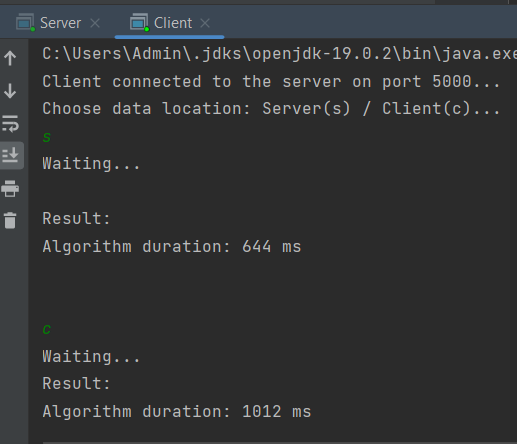


Рисунок 4 – Результати обчислень

***Завдання 2:*** Дослідити швидкість виконання запиту користувача при різних обсягах даних.

*Опис пункту:*

Проведемо заміри швидкості роботи веб-застосунку при різних значеннях величини матриці (500, 1000, 1500, 2000) та кількості потоків (3, 6, 9) та порахуємо прискорення.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Час,ms | Потоки | | | | | | | | |
| Роз. | 3 | | | 6 | | | 9 | | |
| Сер. | Кл. | Пр. | Сер. | Кл. | Пр. | Сер. | Кл. | Пр. |
| 500 | 61 | 108 | 1.77 | 59 | 111 | 1.88 | 52 | 90 | 1.73 |
| 1000 | 852 | 1113 | 1.3 | 754 | 1058 | 1.4 | 731 | 911 | 1.24 |
| 1500 | 9118 | 10120 | 1.11 | 5424 | 7214 | 1.33 | 5806 | 7494 | 1.29 |
| 2000 | 25445 | 31027 | 1.22 | 16154 | 19700 | 1.22 | 16177 | 19203 | 1.19 |

Побудуємо графік прискорення (рисунок 5) за таблицею.

Рисунок 5 – Графік прискорення

***Завдання 3:*** Порівняти реалізацію алгоритму в клієнт-серверній системі та в розподіленій системі з рівноправними процесорами.

*Опис пункту:*

Порівняємо швидкодію алгоритму зі швидкодією попередньої лабораторної роботи (МРІ-методи колективного обміну повідомленнями).

Таблиця замірів 7-ї лабораторної роботи:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Час,ms | Потоки | | | | | | | | |
| Роз. | 3 | | | 6 | | | 9 | | |
| Кол. | Бл. | Пр. | Кол. | Бл. | Пр. | Кол. | Бл. | Пр. |
| 1000 | 1401 | 1803 | 1.28 | 1319 | 1406 | 1.07 | 1324 | 1358 | 1.03 |
| 1500 | 8745 | 11996 | 1.37 | 6355 | 6781 | 1.07 | 6322 | 6654 | 1.05 |
| 2000 | 27356 | 38991 | 1,43 | 17394 | 18532 | 1.06 | 16250 | 17016 | 1.05 |

Таблиця порівняння прискорень:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Час,ms | Потоки | | | | | |
| Роз. | 3 | | 6 | | 9 | |
| Блок.-Кол. | Кл.-Сер. | Блок.-Кол. | Кл.-Сер. | Блок.-Кол. | Кл.-Сер. |
| 1000 | 1.28 | 1.3 | 1.07 | 1.4 | 1.03 | 1.24 |
| 1500 | 1.37 | 1.11 | 1.07 | 1.33 | 1.05 | 1.29 |
| 2000 | 1,43 | 1.22 | 1.06 | 1.22 | 1.05 | 1.19 |

Отже, отримуємо такий результат прискрення (рисунок 6):

Рисунок 6 – Графік прискорення

Та такі графіки швидкодії колективні-сервер (рисунок 7), блокуючі-клієнт (рисунок 7):

Рисунок 7 – Графік швидкодії

Рисунок 8 – Графік швидкодії

Порівнюючи алгоритми можемо зробити висновок, що різниця в прискоренні та швидкодії самих алгоритмів пояснюється затримкою в комунікації між воркерами та клієнтом-сервером. Клієнт-сервер архітектурно здатен комунікувати швидше, а також для комунікації використовувались потоки вводу-виводу а не спеціальні методи MPI, що набагато простіше, а отже і швидше. Данні пересилаються разом, а уже потім розділяються між потоками. Записуються в результуючу матрицю і повертаються разом. Коли в MPI цей процес складніший і на комунікацію затрачається більше часу.

**Висновок:**

Було розроблено веб-застосунокя клієнт-серверної архітектури, яке реалізує алгоритм множення матриць на стороні сервера з використанням паралельних обчислень. Було розглянуто два варіанти реалізації: 1) дані для обчислень знаходяться на сервері та 2) дані для обчислень знаходяться на клієнтській частині застосування. Було проведено дослідження швидкості виконання запиту користувача при різних обсягах даних, а також проведено порівняння алгоритму в клієнт-серверній системі з алгоритмом в розподіленій системі з рівноправними процесорами.

Було проведено експерименти, зібрано статистику результатів роботи алгоритмів за різних вхідних даних та робочих вузлів та кількості потоків, 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/lab8>