Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Комп’ютерний практикум №6**

з дисципліни «Технології паралельних обчислень»

Тема: «Розробка паралельного алгоритму множення матриць з використанням МРІ-методів обміну повідомленнями «один-до-одного» 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 блокуючого та неблокуючого обміну повідомленнями типу point-to-point (див. лекцію та документацію стандарту MPI).

*Опис пункту:*

Подальші два пункти відрізнють використанням методів комунікатора: в блокуючому обміні ми використовуємо методи MPI.COMM\_WORLD.Send та MPI.COMM\_WORLD.Recv, а в неблокуючому використовуємо MPI.COMM\_WORLD.Isend та MPI.COMM\_WORLD.Irecv, а також метод Wait().

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

*Опис пункту та фрагменту коду:*

Задача будується наступним чином:

1. Ініціалізуємо комунікатор, отримуємо розмір комунікатора та індекс поточного процесу.
2. Ділимо код на дві частини, використовуючи умову (if-else) “rank == MASTER”, де константа MASTER дорівнює 0.
3. В частині коду “master” (“роботодавець”, “мастер”) ітераційно розбиваємо матрицю на рівні частини для кожного “worker” (“працівник”, “воркер”), а решту додаємо до роботи останнього воркера.
4. В тій же ітерації відправляємо для кожного воркера його початковий та кінцевий індекс підматриці, саму підматрицю та другу матрицю.
5. В наступному ітераційному процесі ми отримуємо назад від кожного воркера індекси початку та кінція, перемножену підматрицю. Та збираємо їх у результуючу матрицю.
6. В частині коду “worker” кожен працівник отримує свою частину матриці разом з індексами та виконує перемноження підматриці з другою матрицею. Результуючу підматрицюразом з індексами він відправляє роботодавцю, котрий використовує їх як було описано в попередньому пункті.

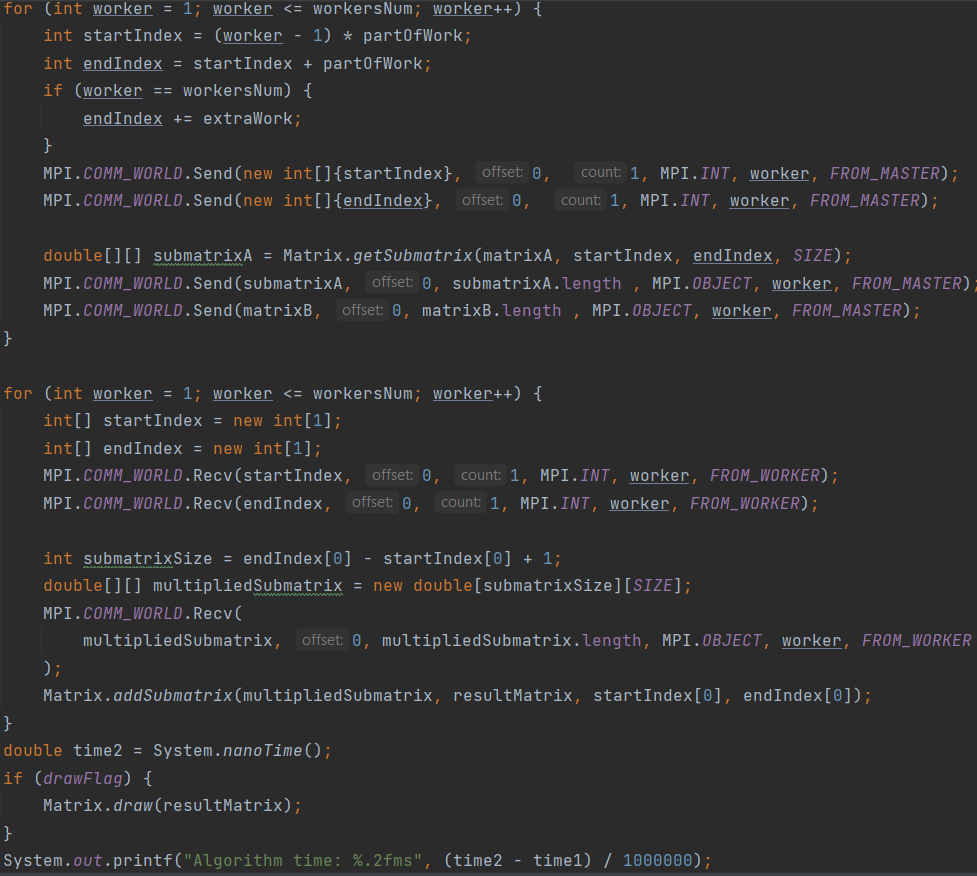


Рисунок 1 – Частина коду “master”

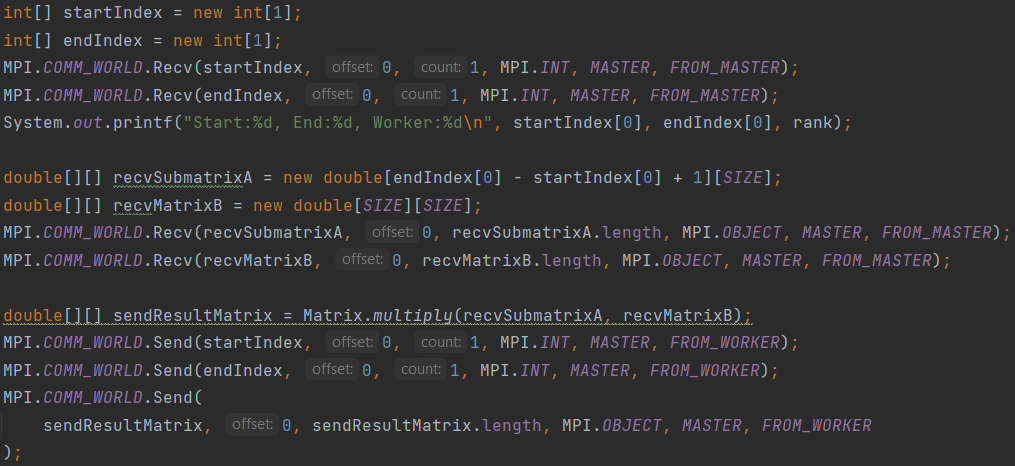


Рисунок 2 – Частина коду “worker”

*Спостереження, результати, висновки:*

Запускаємо код з матрицею зі однаковими значеннями, для перевірки правильності множення. Використовуємо усі потоки процесора (12).

В консолі бачимо частини роботи розділені між 11-ма воркерами 1-им мастером, де 11-тий воркер отримав зверху решту роботи. Також бачимо швидкість обчислення у мілісекундах та виведену матрицю (множення правильне).

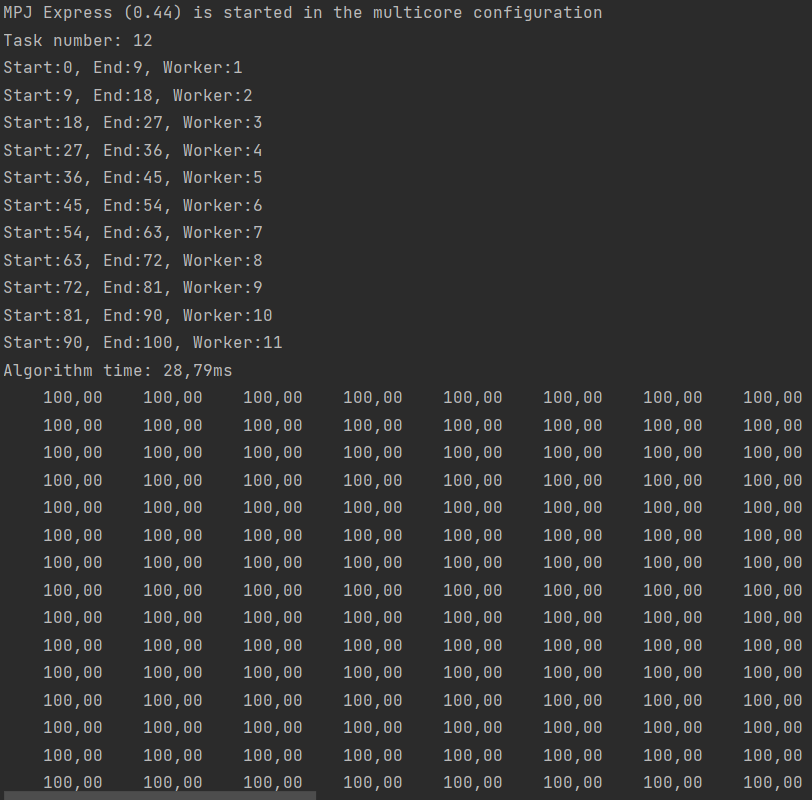


Рисунок 3 – Результати роботи алгоритму множення матриць з використанням методів блокуючих повідомлень

***Завдання 3:*** Реалізувати алгоритм паралельного множення матриць з використанням розподілених обчислень в MPI з використанням методів неблокуючого обміну повідомленнями.

*Опис пункту та фрагменту коду:*

Як було згадано у першому пункті, завдання відрізняється використанням методів MPI.COMM\_WORLD.Isend та MPI.COMM\_WORLD.Irecv, а також методу Wait(). Решта структури коду одинакова, окрім того, що ми помічаємо кожен “Request” для виконання методу Wait().

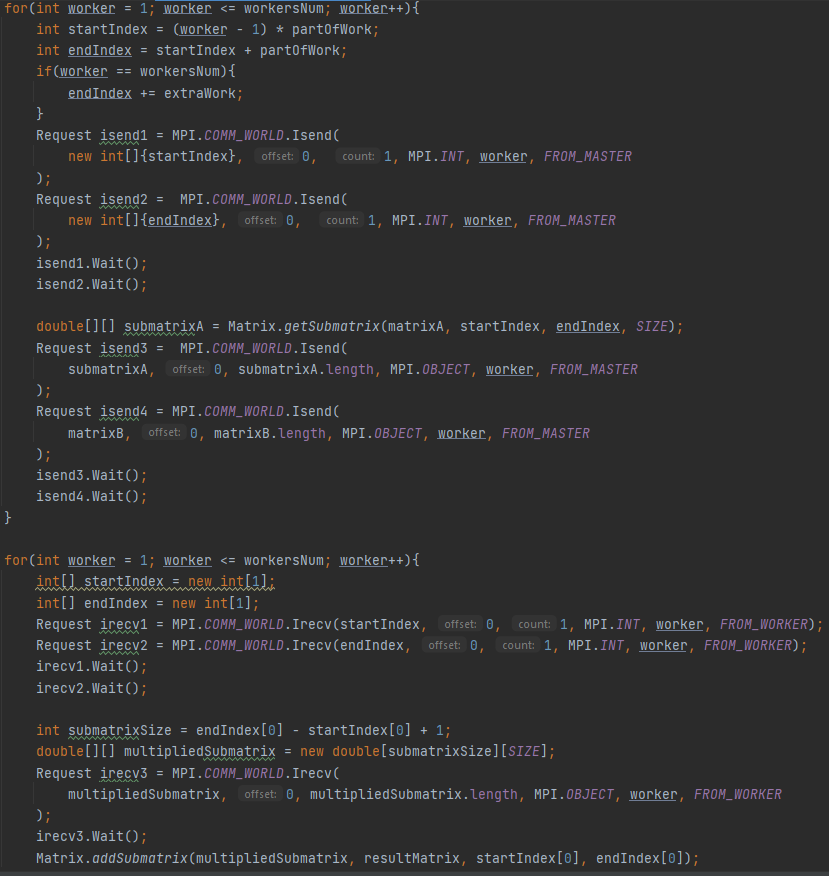


Рисунок 4 – Частина коду “master”

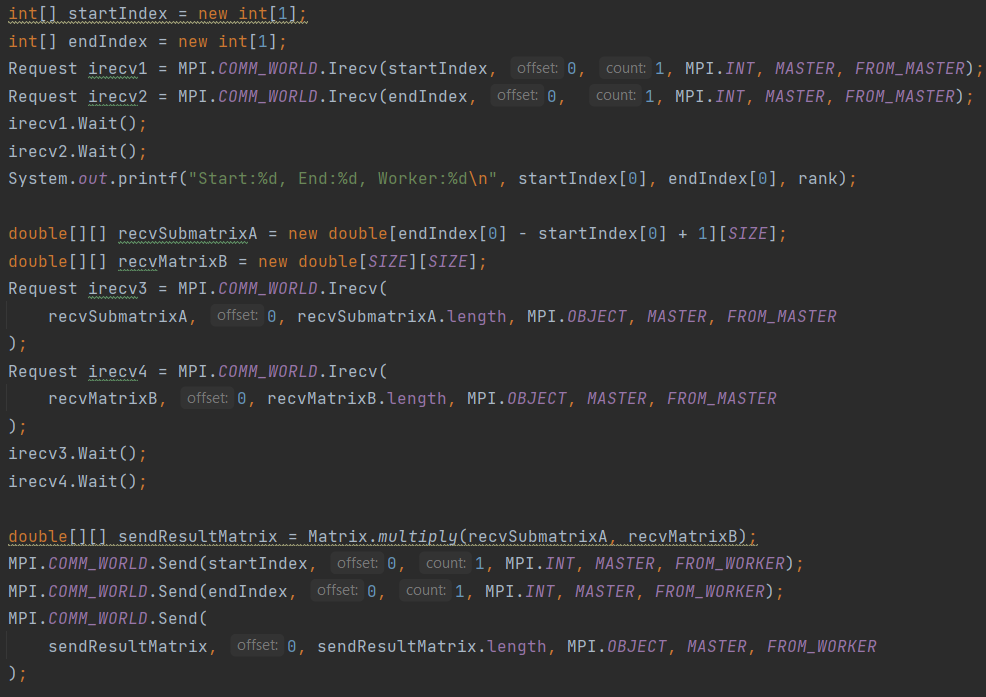


Рисунок 5 – Частина коду “worker”

*Спостереження, результати, висновки:*

Бачимо послідовний вивід інформації про роботу імітаційного моделювання на кожній ітерації. 1001 ітерація зроблена для прикладу статусу “terminated” після виконання методу awaitTermination().

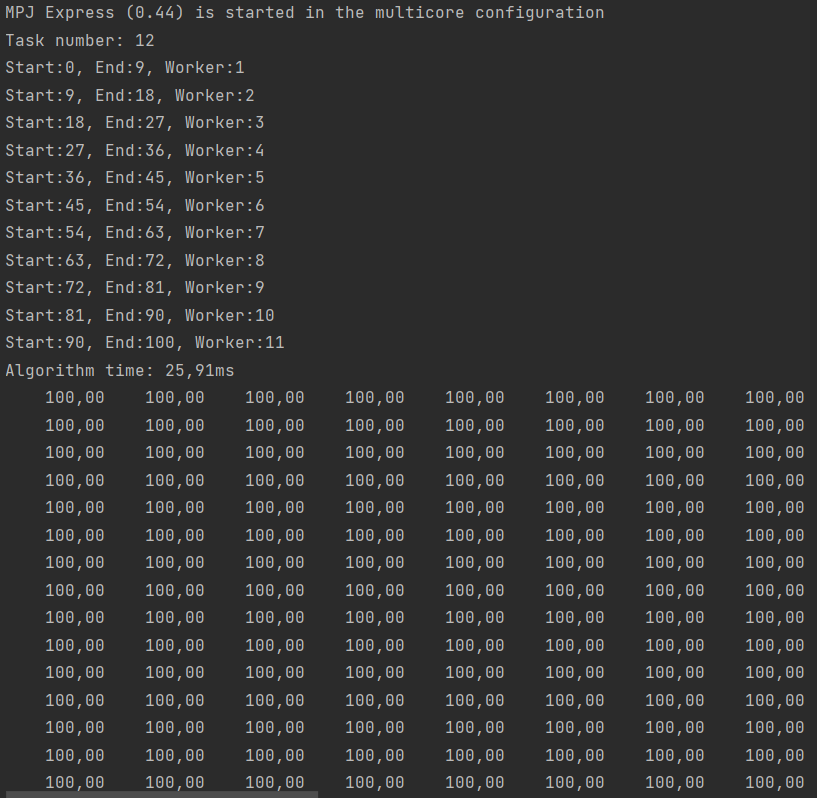


Рисунок 6 – Результати роботи алгоритму множення матриць з використанням методів неблокуючих повідомлень

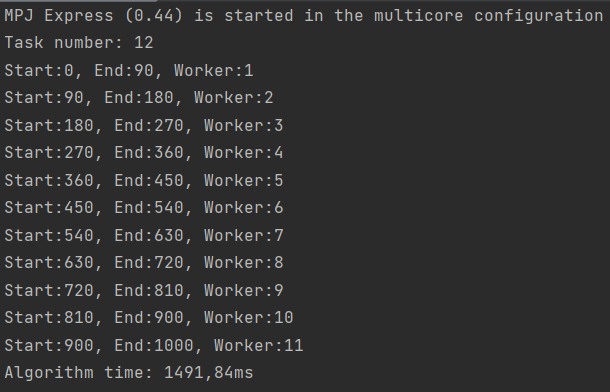
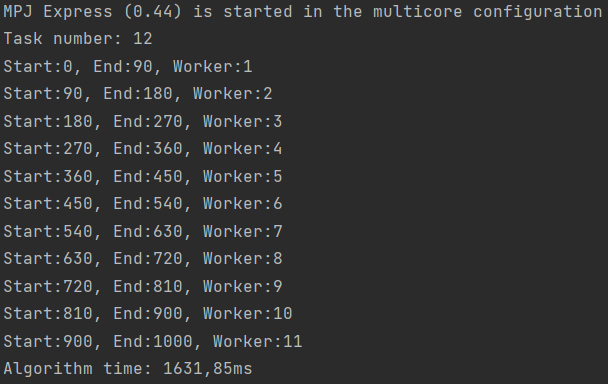


Рисунок 7 – Різниця швидкості блокуючого та неблокуючого обміну повідомленнями

***Завдання 4:*** 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 блокуючих та неблокуючих методів обміну повідомленнями.

*Опис пункту, спостереження, результати, висновки:*

Для виконання завдання, ми будемо змінювати розмір комунікатора, змінюючи значення параметрів запуску від 3 до 12, бо має бути мініму один мастер та два воркера, а також процесор на якому виконується обчислення має 12 потоків. Також будемо варіювати розмір матриці від 100 до 1000.

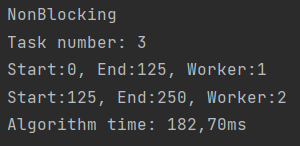
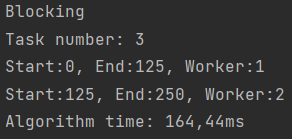


Рисунок 8 – Blocking - NonBlocking, Task - 3, Matrix - 250

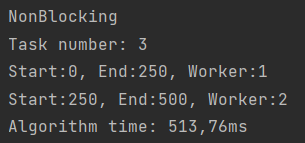
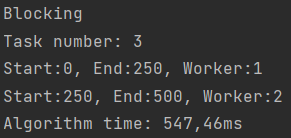


Рисунок 9 – Blocking - NonBlocking, Task - 3, Matrix - 500

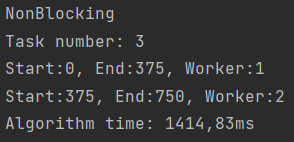
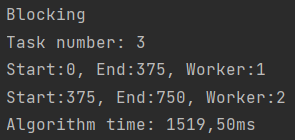


Рисунок 10 – Blocking - NonBlocking, Task - 3, Matrix - 750

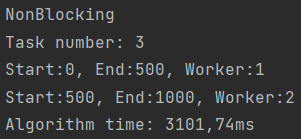
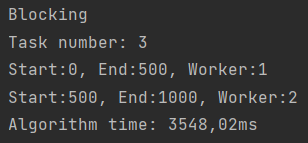


Рисунок 11 – Blocking - NonBlocking, Task - 3, Matrix - 1000

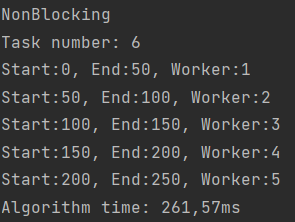
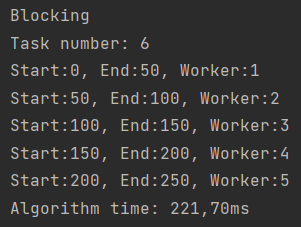


Рисунок 12 – Blocking - NonBlocking, Task - 6, Matrix - 250

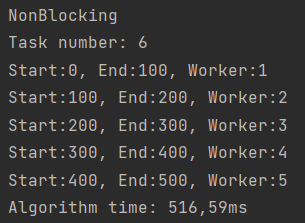
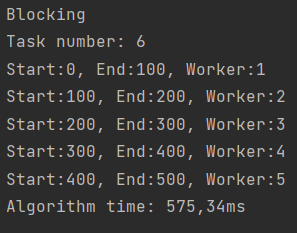


Рисунок 13 – Blocking - NonBlocking, Task - 6, Matrix - 500

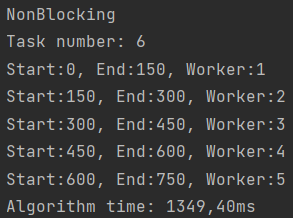
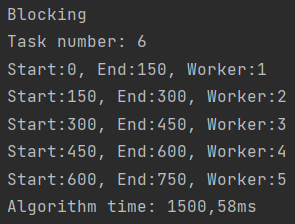


Рисунок 14 – Blocking - NonBlocking, Task - 6, Matrix - 750

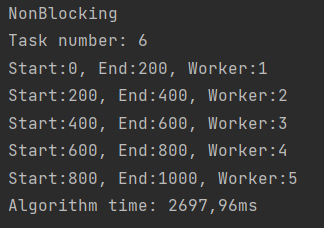
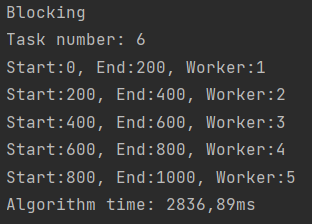


Рисунок 15 – Blocking - NonBlocking, Task - 6, Matrix - 1000

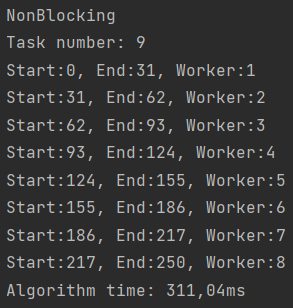
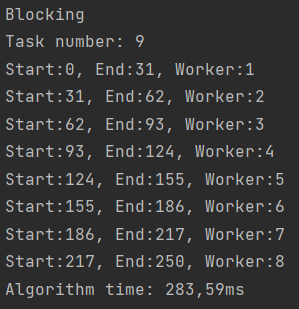


Рисунок 16 – Blocking - NonBlocking, Task - 9, Matrix - 250

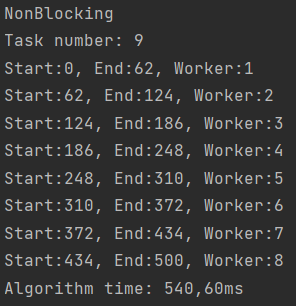
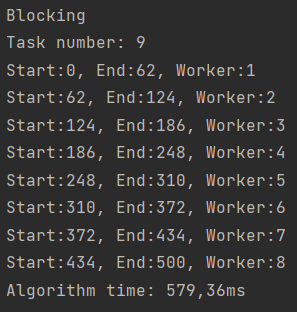


Рисунок 17 – Blocking - NonBlocking, Task - 9, Matrix - 500

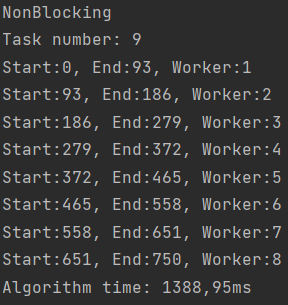
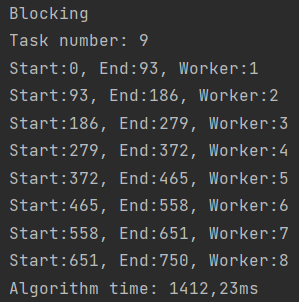


Рисунок 18 – Blocking - NonBlocking, Task - 9, Matrix - 750

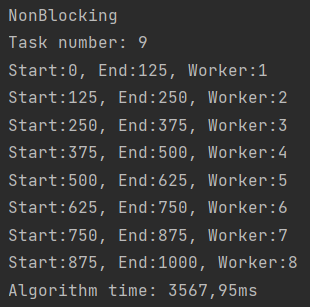
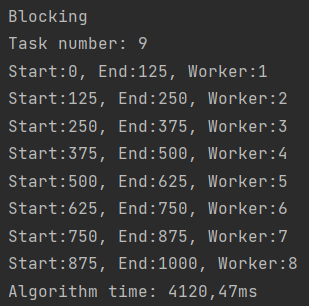


Рисунок 19 – Blocking - NonBlocking, Task - 9, Matrix - 1000

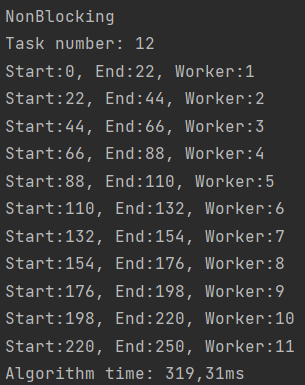
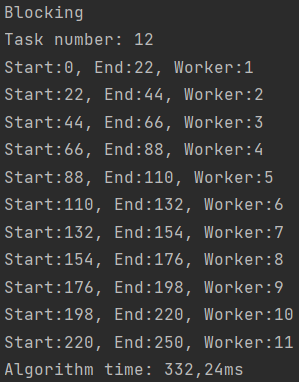


Рисунок 20 – Blocking - NonBlocking, Task - 12, Matrix - 250

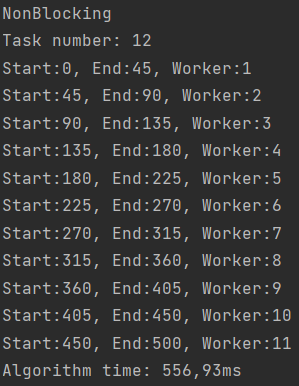
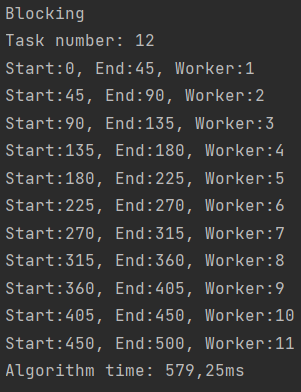


Рисунок 21 – Blocking - NonBlocking, Task - 12, Matrix - 500

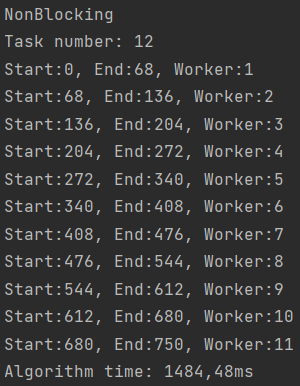
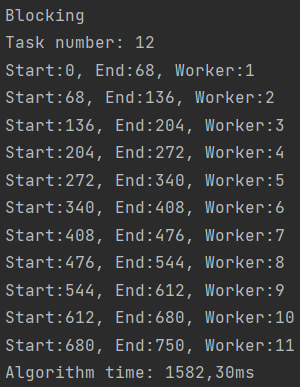


Рисунок 22 – Blocking - NonBlocking, Task - 12, Matrix - 750

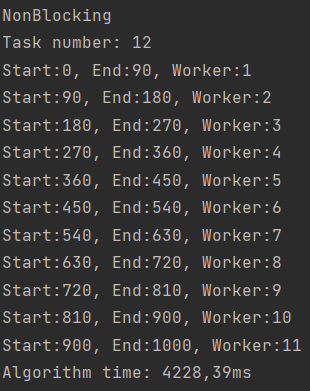
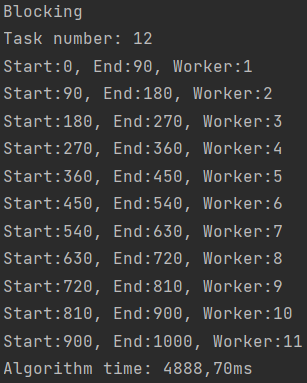


Рисунок 23 – Blocking - NonBlocking, Task - 12, Matrix - 1000

Рисунок 24 – Графік Task - 3

Рисунок 25 – Графік Task - 6

Рисунок 26 – Графік Task - 9

Рисунок 27 – Графік Task - 12

Рисунок 28 – Графік Size - 250

Рисунок 29 – Графік Size - 500

Рисунок 30 – Графік Size - 750

Рисунок 31 – Графік Size - 1000

Висновок:

Як і очікувалось, в цілому, паралельні обчислення з неблокуючими методами обміну повідомленнями виконуються швидше ніж з блокуючими. Різниця більш відчутна на більших розмірах матриць.

Збільшення кількості вузлів впливає по різному на різних величинах матриці. Так бачимо, що на графіках з розмірами матриць 250, 500, 750 та 1000, графіки мають різні форми, і досягають максимальної ефективності на різній кількості вузлів. 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-одного” з використанням блокуючих та неблокуючих 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/lab6>