

Міністерство освіти і науки України
Національний технічний університет України «Київський
політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 2 з дисципліни
«Технології паралельних обчислень»

«Розробка паралельних алгоритмів множення матриць та дослідження їх
ефективності»

Виконав(ла)

ІП-14 Сергієнко Ю. В.
(шифр, прізвище, ім'я, по батькові)

Перевірив

Дифучина О. Ю.
(прізвище, ім'я, по батькові)

Київ 2024

Комп'ютерний практикум 2

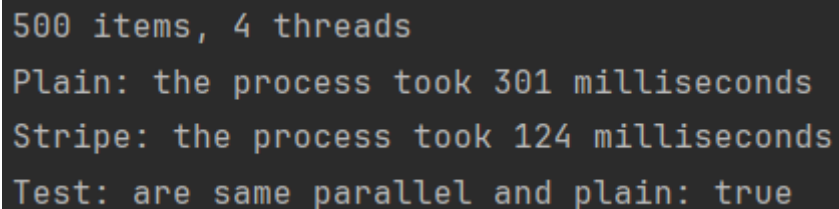
Тема: Розробка паралельних алгоритмів множення матриць та дослідження їх ефективності.

Виконання:

1. Реалізуйте стрічковий алгоритм множення матриць. Результат множення записуйте в об'єкт класу `Result`. **30 балів.**

Було розроблено стрічковий алгоритм типу елемент – ітерація, тобто за проходження ітерації потоків, було обраховано один результуючий елемент. Даний алгоритм вирішує основну проблему багатопотокових програм (`memory consistency error`) – потоки звертаються до різних об'єктів матриці, адже вони працюють з окремими рядками.

Було розроблено методи `generate()` та `initialize()` класу `Generator` для полегшення генерації початкових даних для матриць. Також було створено клас `Matrix` (аналогія `Result`) для збереження результату обчислень. Для перевірки роботи алгоритму було додано звичайний алгоритм множення матриць. Результат виконання програми бачимо на рисунку 1.



```
500 items, 4 threads
Plain: the process took 301 milliseconds
Stripe: the process took 124 milliseconds
Test: are same parallel and plain: true
```

Рисунок 1 – Виконання стрічкового алгоритму

2. Реалізуйте алгоритм Фокса множення матриць. **30 балів.**

Алгоритм представляє собою паралельне множення блоків матриці, де кількість блоків визначається кількістю потоків. Для полегшення роботи з матрицею було створено метод `breakIntoBlocks()`, за допомогою якого можна створити матрицю об'єктів `Matrix` (`Matrix[][]`). Далі було виконано множення матриць за допомогою алгоритму Фокса. Для цього були дописані допоміжні методи `add()` та `multiply()` для `Matrix` об'єктів. Було збережено принцип множення матриць, а саме: $C[i][j] = C[i][j] + A[i][i+1] * B[i+1][j]$

$B[i+1][j]$. Після того, як ми отримали результат множення матриць, його було зкомбіновано в один об'єкт матриці за допомогою методу *combine()*. Результат виконання програми можемо бачити на рисунку 2.

```
500 items, 4 threads
Plain: the process took 209 milliseconds
Stripe: the process took 79 milliseconds
Fox: the process took 87 milliseconds
Test: are same parallel and plain: true
Test: are same fox and plain: true
```

Рисунок 2 – Виконання алгоритму Фокса

3. Виконайте експерименти, варіюючи розмірність матриць, які перемножуються, для обох алгоритмів, та реєструючи час виконання алгоритму. Порівняйте результати дослідження ефективності обох алгоритмів. **20 балів.** Виконайте експерименти, варіюючи кількість потоків, що використовується для паралельного множення матриць, та реєструючи час виконання. Порівняйте результати дослідження ефективності обох алгоритмів. **20 балів.**

Було виконано експерименти з дослідження часу виконання алгоритмів у залежності від розмірів матриць та кількості потоків. Результати дослідів для стрічкового алгоритму та алгоритму Фокса відображені у таблиці 1 та 2 відповідно.

Таблиця 1 – Результати експериментів над стрічковим алгоритмом

Matrix size	Serial, s	Stripe, s		
		4 Threads	9 Threads	16 Threads
500	0.192	0.084	0.088	0.114
1000	1.95	0.51	0.35	0.3
1500	9.4	2.5	2.2	2.1
2000	33.9	9.2	7.4	7
2500	78.9	21.1	15.6	14.9

Продовження таблиці 1:

Matrix size	Serial, s	Stripe, s		
		4 Threads	9 Threads	16 Threads
3000	158.5	45.2	32.7	30.8

Таблиця 2 – Результати експериментів над алгоритмом Фокса

Matrix size	Serial, s	Fox, s		
		4 Threads	9 Threads	16 Threads
500	0.2	0.095	0.074	0.082
1000	1.95	0.37	0.31	0.28
1500	9.4	1.7	1.1	0.8
2000	33.9	4.4	2.5	2.2
2500	78.9	12	7	5.6
3000	158.5	16.9	11.4	11.6

Прискорення алгоритмів відображено у таблиці 3.

Таблиця 3 – Прискорення алгоритмів множення

Matrix size	4 Threads		9 Threads		16 Threads	
	Stripe, t	Fox, t	Stripe, t	Fox, t	Stripe, t	Fox, t
500	2.23	2.11	2.18	2.7	1.68	2.44
1000	3.8	5.2	5.6	6.3	6.5	6.96
1500	3.76	5.53	4.27	8.5	4.48	11.75
2000	3.68	7.7	4.58	13.56	4.8	15.4
2500	3.74	6.58	5.06	11.2	5.3	14
3000	3.5	9.38	4.8	13.9	5.15	13.66

Результати вимірів відображені на рисунках 3, 4 та 5.

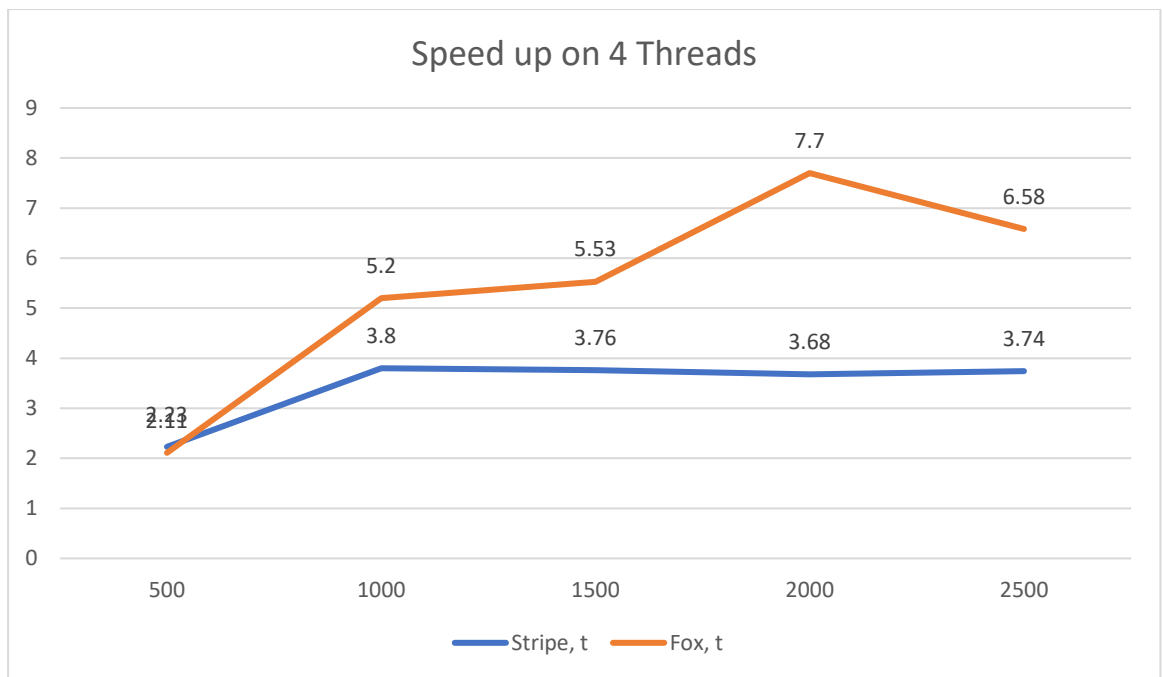


Рисунок 3 – Прискорення при 4 потоках

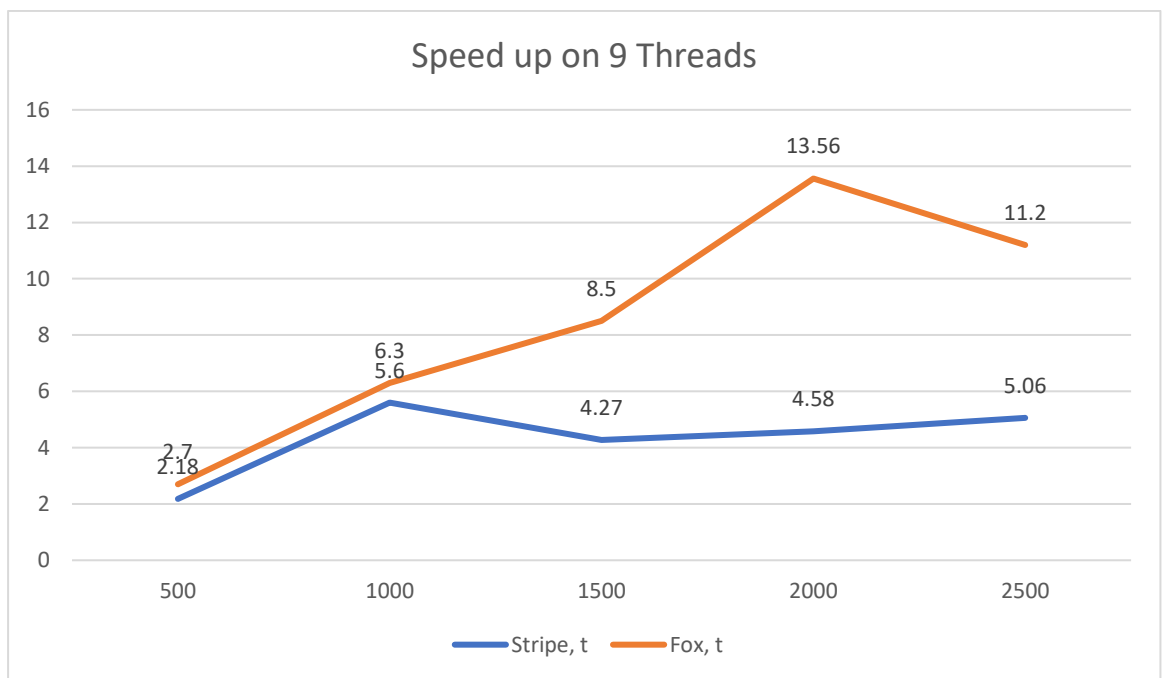


Рисунок 4 – Прискорення при 9 потоках

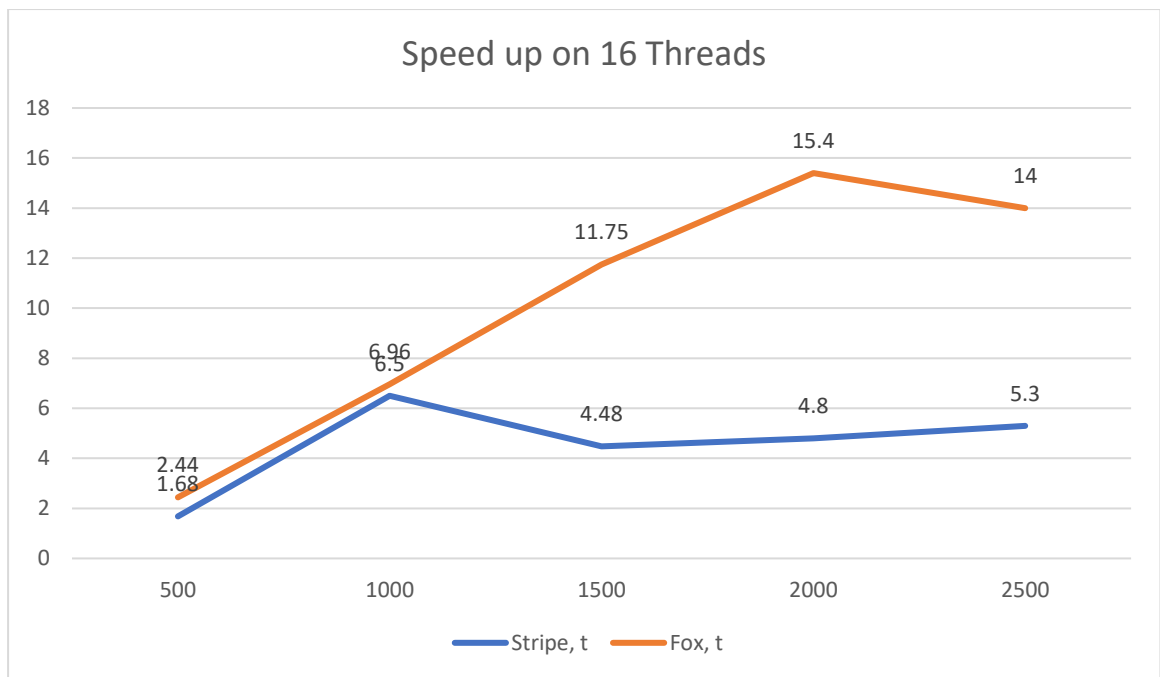


Рисунок 5 – Прискорення при 16 потоках

Висновок

Під час виконання даного комп'ютерного практикуму я набув знань та навичок щодо розробки паралельних алгоритмів, в моєму випадку – на прикладі множення матриць. Було розроблено стрічковий алгоритм множення, в основі якого є використання потоками різних рядків одночасно. Також було розроблено рішення для алгоритму множення Фокса: для підготовки матриці її було розбито на менші частини відповідно до кількості потоків, далі помножено блоками та зібрано в один об'єкт.

Було виконано експерименти для дослідження ефективності алгоритмів відносно звичайного.

Код програми доступний на [Github](#).