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

# Лабораторна робота 1

**«Програмування потоків»**

з дисципліни

«Програмне забезпечення високопродуктивних комп’ютерних систем»

Виконав
студент групи ІМ-13
Кравчук Ілля Володимирович
номер у списку групи: 9

Перевірив:
доц. Корочкін О. В.

**Київ 2024**

**Лабораторна робота ЛР1.3.**

**ПОТОКИ В МОВІ JAVA**

Завдання: Створити програму з паралельними потоками, що реалізують функції F1, F2 та F3.

Варіант 9

F1: 1.20 D = MIN(A + B) * (B + C) *(MA*MD)

F2: 2.28 MF = MIN(MH)*MK*ML

F3: S = V*MO + R*MP + P*(MO*MS)

## Виконання лабораторної роботи

Лабораторну роботу було виконано на мові Java через її потужну підтримку паралельного програмування за допомогою потоків (threads). Вона надає різні інструменти для створення та управління потоками. Підтримка паралельних процесів у Java полягає в її вбудованій підтримці програмування паралельних процесів з використанням потоків. Клас Thread та інтерфейс Runnable в Java дозволяють створювати потоки, надаючи гнучкість у виборі способу реалізації потоків. Використання методу join() у мові Java дозволяє забезпечити синхронізацію основного потоку з іншими потоками.

Необхідно було реалізувати налагодження паралельної програми та дослідити її виконання для малих та великих значень N. Для невеликого значення N<9 введення в потоці векторів та матриць здійсняються за допомогою клавіатуру, а для великих значень N =1000 для введення даних здійснюється двома шляхами це генерація випадкових чисел, або заповнення спеціальним числами для кожного потоку.

## Проблеми з вводом/виводом в програмі

Проблема, яка виникла була пов’язана з вводом даних при N<9(Рис.1). При спробі декількох потоків одночасно отримати введення з клавіатури можуть виникнути конфлікти та блокування. Суть проблеми полягає в тому ,що після вводу N запускаються три потоки, і всі три мають спільний доступ до ресурсу. Способи вирішення цієї проблеми це використання “семафорів” та “локів”.

Проблема з виводом результатів пов’язана з тим, що кожен потік після свого виконання виводить результат. Однак інший потік може навіть не отримати даних для виконання своєї функції. Цю проблему можна вирішити шляхом запису результатів у функцію, яка буде виводити результат всіх потоків після завершення останнього потоку.

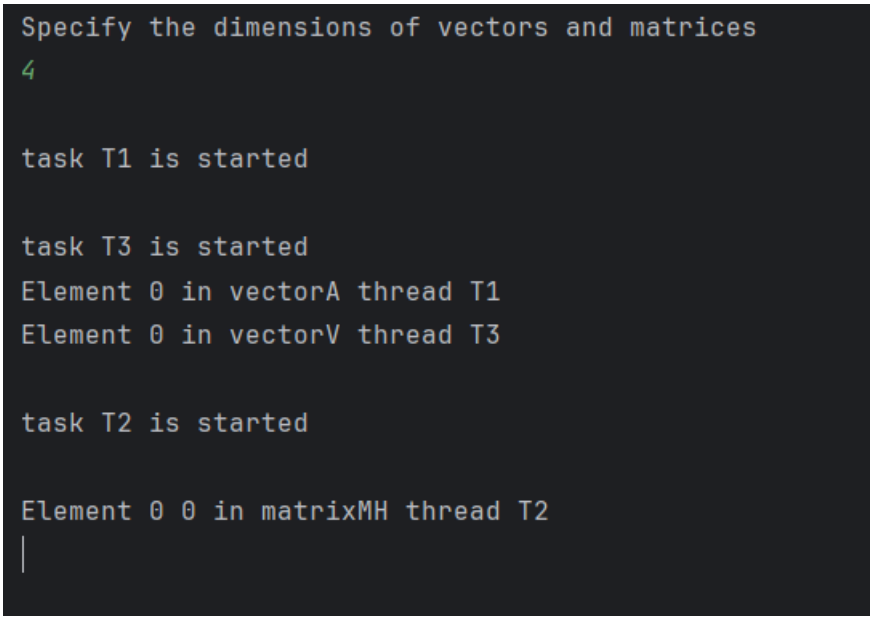



Рис.1

Якщо значення N=1000 вводиться, то дані будуть записані у файл(Рис.2). В програмі передбачено можливість вибору методу заповнення матриць, векторів: спеціальними значеннями для кожного потоку або випадковими числами. Діапазон випадкових чисел становить від -100 до 100 з кроком 0.1.

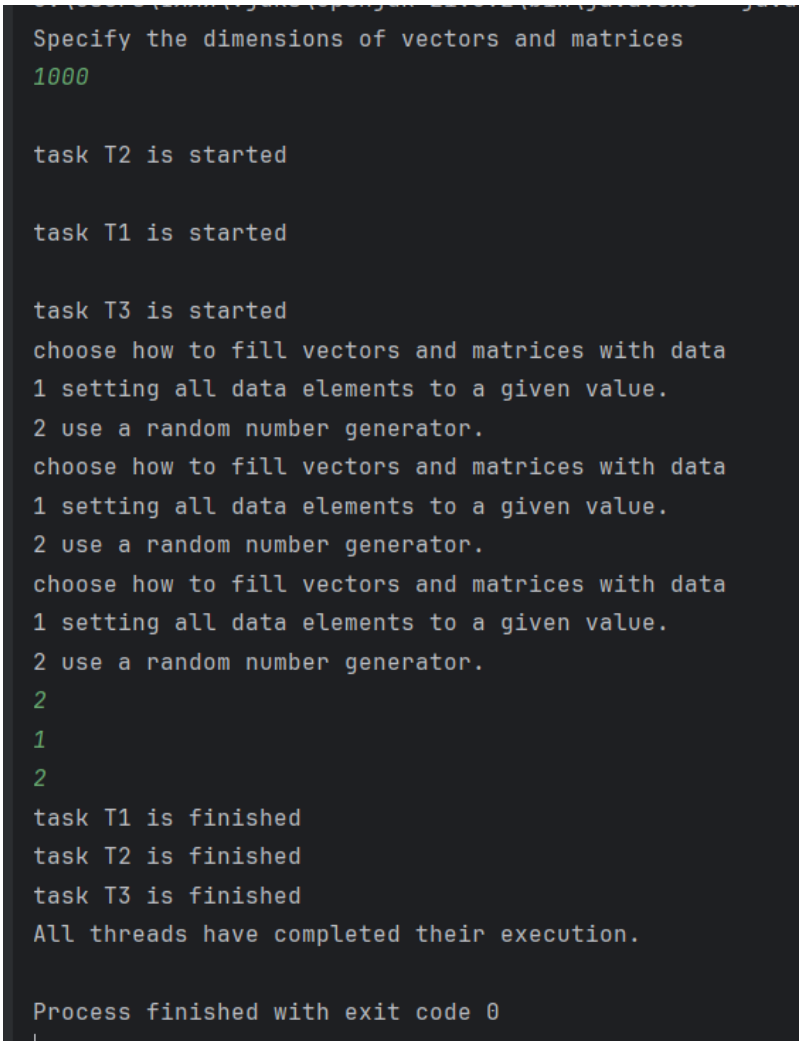



Рис.2

Програма успішно виконалась і всіх дані і результати були записані у файли(Рис.3, Рис.4, Рис.5 ). Знову виникла проблема з вводом даних у потоки.

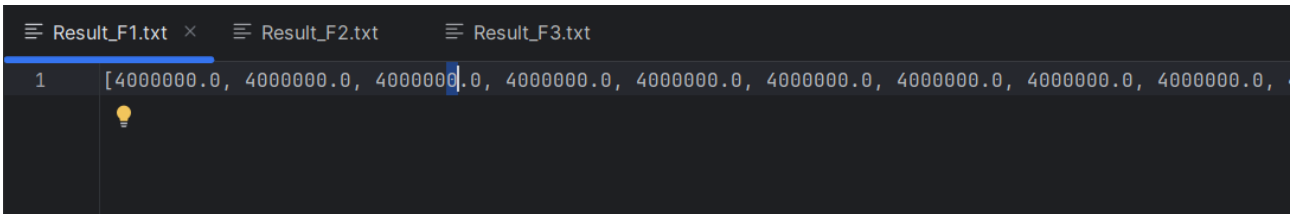



Рис.3

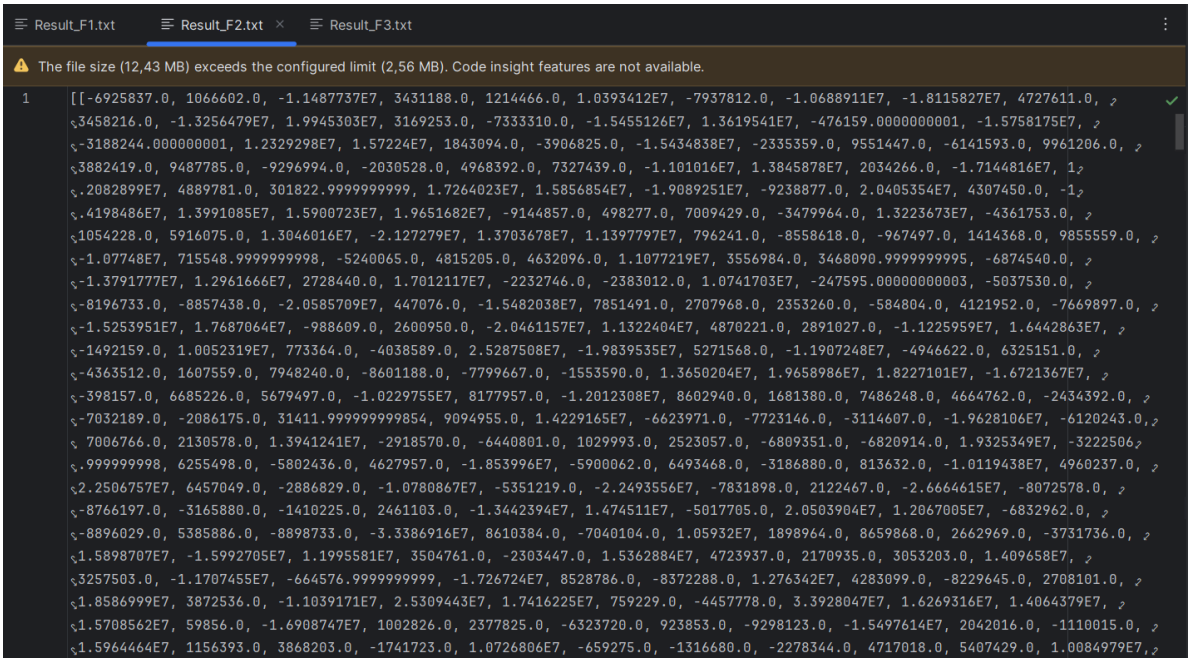



Рис.4

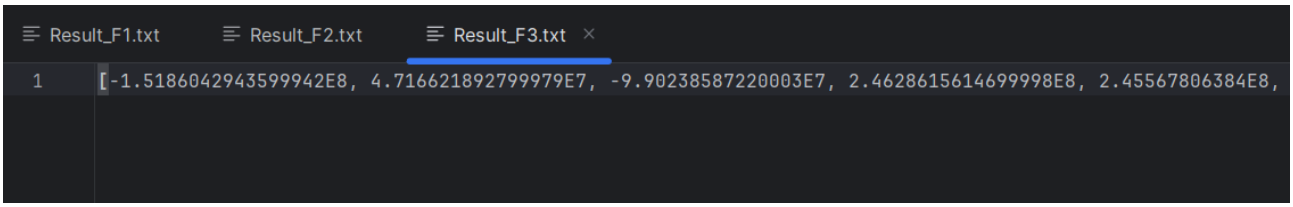


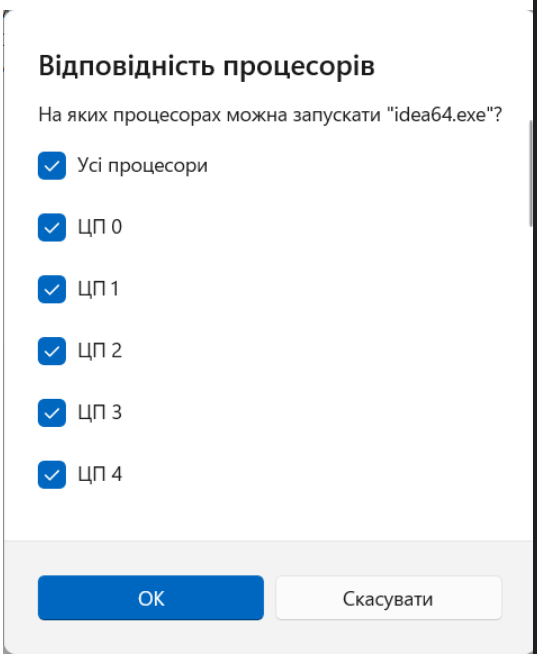

Рис.5

## Дослідження завантаженості ядер процесора

В ноутбуці є 14 ядер і 20 логічних процесорів(Рис.6).


Рис.6

Результати виконання програми на всіх логічних процесорах:




```
Specify the dimensions of vectors and matrices
1000

task T1 is started

task T3 is started

task T2 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.
2
2
2
task T2 is finished
task T1 is finished
task T3 is finished
All threads have completed their execution.
program execution time = 6425

Process finished with exit code 0
```

Рис.7 Рис.8

Час виконання програми на всіх ядрах процесора 6,4 секунд(Рис.7 ,Рис.8).

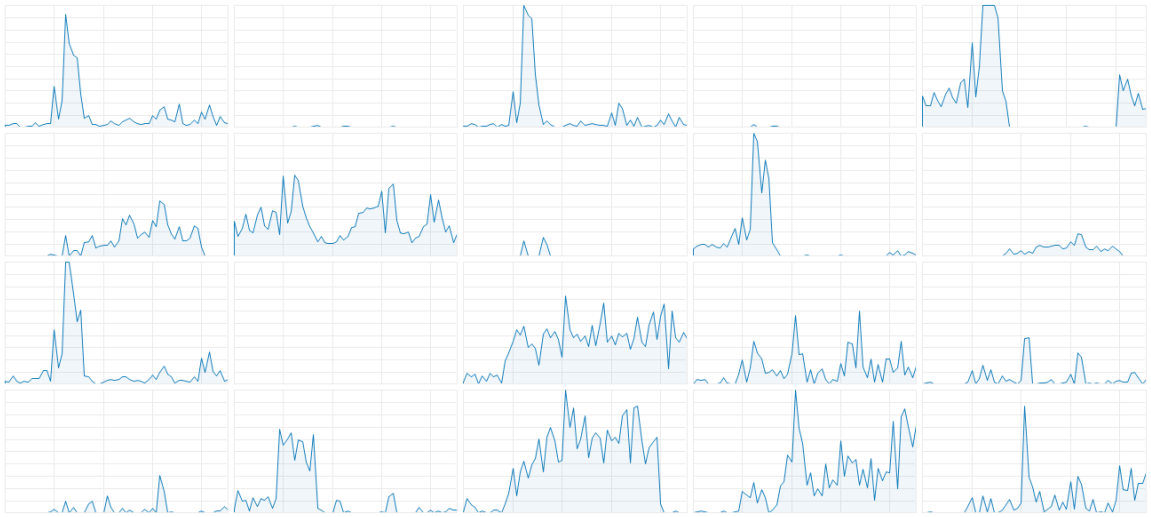
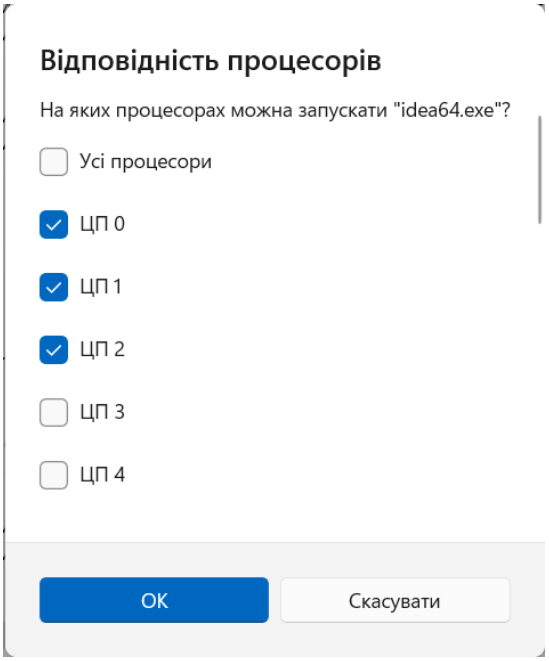


Рис.9(завантаженість всіх ядер)

Результати виконання програми на **3** логічних процесорах :




```
Specify the dimensions of vectors and matrices
1000

task T2 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.

task T1 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.

task T3 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.
2
2
2
task T3 is finished
task T2 is finished
task T1 is finished
All threads have completed their execution.
program execution time = 8953

Process finished with exit code 0
```

Рис.10 Рис.11

Час виконання програми на трьох ядрах процесора 9 секунд(Рис.10, Рис.11).

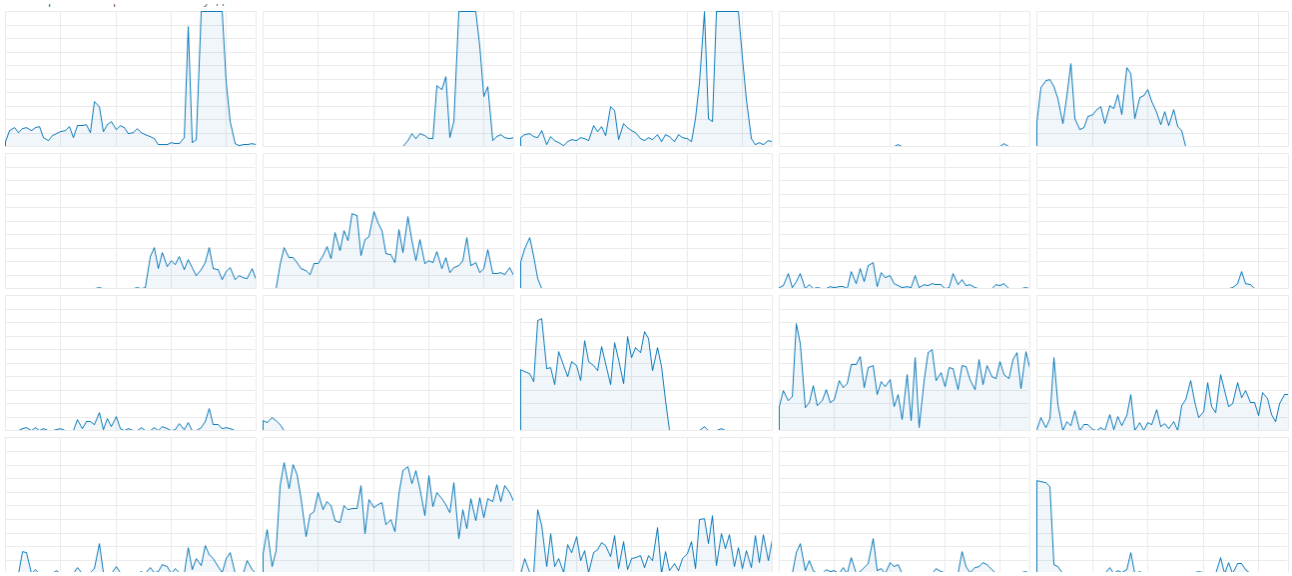
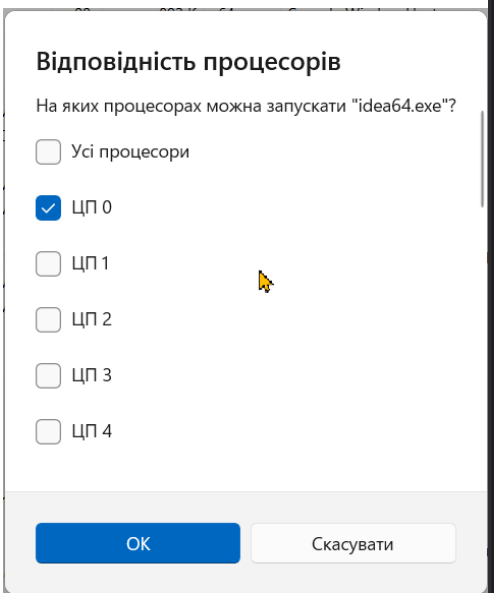


Рис.12(завантаженість 3 ядер)

Після зменшення ядер до 3 час виконання програми збільшилось на 2,6 секунд порівняно з використанням всіх ядер для виконання програми. Також можна спостерігати рівномірну навантаженість між усіма задіяними ядрами у процесі виконання програми(Рис.9, Рис.12).

Результати виконання програми на **1** логічному процесорі:




```
Specify the dimensions of vectors and matrices
1000

task T1 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.

task T2 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.

task T3 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.
2
2
2
task T3 is finished
task T1 is finished
task T2 is finished
All threads have completed their execution.
program execution time = 13492

Process finished with exit code 0
```

Рис.13 Рис.14

Час виконання програми на одному ядрі процесора 13,5 секунд(Рис.13, Рис.14).

Результати виконання програми на 4 логічних процесорах:

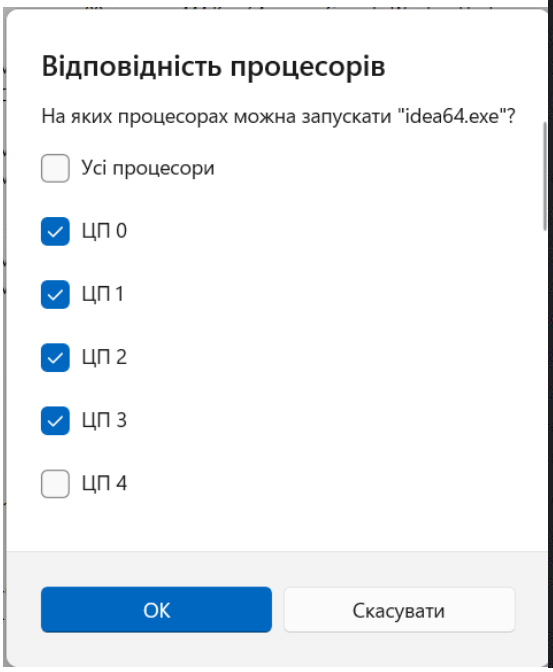



Рис.15

```
Specify the dimensions of vectors and matrices
1000

task T1 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.

task T2 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.

task T3 is started
choose how to fill vectors and matrices with data
1 setting all data elements to a given value.
2 use a random number generator.
2
2
2
task T1 is finished
task T3 is finished
task T2 is finished
All threads have completed their execution.
program execution time = 8425

Process finished with exit code 0
```

Рис.16

Час виконання програми на чотирьох ядрах процесора 8,4 секунд(Рис.15, Рис.16).

$К_у = T_1 / T_4 = 13.5 / 8.4 \approx 1.61$

Аналізуючи завантаження ядер(Рис.9, Рис.12) до та після запуску програми, можна зробити висновок, що програма ефективно розподіляє ресурси обчислювальної системи для досягнення оптимальної продуктивності. При обмеженні кількості потоків операційна система автоматично розподіляє навантаження між доступними ядрами, що забезпечує більш ефективне використання ресурсів процесора.

# Висновок

1. Під час виконання розробки програми на мові Java, а саме створення потоків за допомогою наслідування класу Thread. У головному потоці main використовувалися методи start() для активації потоків T1-T3 та join() для очікування завершення їх виконання.
2. Під час виконання лабораторної роботи виникла проблема з введенням даних, оскільки кожен потік мав доступ до спільного ресурсу, що викликало конфлікт вводу даних. Вирішення цієї проблеми це використання механізмів синхронізації.
3. Проблема з виведенням результатів у програмі виникає через асинхронне завершення потоків, що призводить до несинхронізованого виведення даних. Це можна вирішити за допомогою функції, яка прийматиме результати всіх потоків, а після завершення останнього виводитиме всі результати.
4. У процесі дослідження завантаженості ядер процесора, було з’ясовано таку закономірніть. При збільшені кількості ядер процесора, швидкість виконання програми збільшується, оскільки система може ефективно розподіляти завдання між ними. Також отриманий у процесі дослідження коефіцієнт прискорення(1,61) показує ефективність використання паралельних потоків у програмі.