НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Звіти до комп’ютерних практикумів з кредитного модуля

“Parallel computing”

**Виконав**

**Студенти групи ІТ-04**

**Гавриленко Я.С**

**Перевірив:**

Київ – 2022

**Комп‘ютерний практикум No 4**

**Розробка паралельних програм з використанням пулів потоків, екзекьюторів та ForkJoinFramework**

ЗАВДАННЯ

Зображення, що містить текст

Автоматично згенерований опис

Зображення, що містить текст

Автоматично згенерований опис

**ВИКОНАННЯ**

Повний код лабораторної: <https://github.com/yan14171/parralel_calcullation_labs/tree/master/lab>4

**Завдання 1**

Для документу створимо рекурсивну таску, яка почне виконання з повним контентом файлу. Під час виконання, будемо розділяти файл та для кожної половини створювати нову таску, якщо довжина переданого тексту буде більша за певний ліміт.

Результати замірів прискорення відносно лінійного алгоритму, що є основним показником ефективності, можна побачити на рисунку 1.1. Результати дослідження є середнім арифметичним із п’яти прогонів кожної функції підрахунку.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.1 Результат замірів прискорення

Як бачимо, результат при використанні паралелізації за допомогою пулу ForkJoin у декілька разів кращий. Також зазначимо, що дана оптимізація має сенс лише при масивах даних великого розміру.

**Завдання 2**

Для виконання завдання 2 був обраний алгоритм множення матриць стрічковим методом. Так, при передачі у таску стрічки розміром більшої за певний ліміт, створюється дві рекурсивні таски, в які передаються половини початкової стрічки. Потім результати сумуються і передаються далі. Результати дослідження є середнім арифметичним із п’яти прогонів кожної функції підрахунку.

Результати роботи оригінального та нового стрічкового прикладу можна побачити на рисунку 1.2.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.2 Реалізація стрічкового прикладу

У даному випадку прискорення буде дорівнювати

,

де S – прискорення, Tseq – час виконання лінійного алгоритму, Tpar – час виконання паралельного.

**Завдання 3**

Реалізуємо поставлену задачу, використовуючи розділення документу на частини із завдання 1. Також, будемо створювати окрему таску під кожен документ, який треба порівняти. Додаючи слова до сету, будемо знаходити спільні слова.

Вміст файлів можна побачити на рисунку 1.3.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.3 Вміст вхідних файлів

На рисунку 1.4 показан результат виконання програми.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.4 Результат виконання програми

**Завдання 4**

Реалізація завдання 4 майже повінстю повторює реалізацію завдання 3. Однак, для збереження інформації про факт знаходження ключового слова у тому чи іншому файлі, потрібно було відмовитися від розділення документів на підзадачі за лімітом довжини

Вміст файлів можна побачити на рисунку 1.5.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.5 Вміст вхідних файлів

На рисунку 1.6 показан результат виконання програми.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.6 Результат виконання програми

**ВИСНОВОК**

В рамках роботи, ми на практиці познайомились з пулом потоків ForkJoin, його застосуванням, перевагами та недоліками.

Маючи декілька задач, ми змогли як покращити час виконання раніше реалізованих алгоритмів, за рахунок використання цього пулу потоків, так і створити оптимальні версії інших алгоритмів, задачі яких сприятливі для цього.