Міністерство освіти і науки України

Національний технічний університет України «КПІ» імені Ігоря Сікорського

Кафедра інформаційних систем та технологій ФІОТ

ЗВІТ

з лабораторної роботи №1

з навчальної дисципліни «Методи та технології паралельного програмування»

Тема: Засоби створення та керування потоками в паралельних мультипоточних програмах

Виконав:

Студент 5 курсу кафедри ІСТ ФІОТ,

Навчальної групи ІК-11мп

Клімов В.В.

Київ 2022

Завдання

Реалізувати послідовну та паралельну мультипоточну обробку незалежних задач (наприклад, parameter sweep – вирішуються екземпляри однієї задачі для різних значень параметрів). Реалізувати приклади задач трьох типів:

1. CPU-bound – складні обчислення з невеликим обсягом даних
2. Memory-bound – робота з даними, що зберігаються в пам’яті
3. IO-bound – робота з даними на диску.

Виміряти залежність часу виконання від кількості потоків.

Результати виконання роботи

Програму було написано з використанням мови C++, стандартної бібліотеки std::thread.

Задача CPU-bound

Реалізовано функцію «foo», що виконує велику кількість обчислень в одному потоці та навантажує процесор. Навантаження на одному потоці 19%.

Також реалізовано функцію «foo_c», що виконується на 8 потоках та виконує ті ж самі обчислення. Навантаження на процесор сягає 54%. Це обумовлено тим, що усі ядра процесору задіяні під час виконання обчислень.

Час виконання в одному потоці – 3086 мілісекунд.

Час виконання на 8 потоках – 634 мілісекунди.

Задача Memory-bound

Реалізовано функцію «recursive_Fibonacci», що виконує обчислення послідовності Фібоначчі в одному потоці. Навантаження на оперативну пам’ять відбувається за рахунок створення великої кількості тимчасових значень, що зберігаються в стеку.

Також реалізовано функцію «recursive_Fibonacci_c», що виконується на 8 потоках та виконує ті ж самі обчислення.

Час виконання в одному потоці – 5872 мілісекунд.

Час виконання на 8 потоках – 1656 мілісекунд.

Задача IO bound

Реалізовано функцію «read», що виконує зчитування з файлу, що зберігається на жорсткому диску, великої кількості даних.

Також реалізовано функцію «r», що виконує функцію «read» на 8 потоках (кожний виклик функції «read» зчитує 1/8 файлу.

Час виконання в одному потоці – 104 мілісекунд.

Час виконання на 8 потоках – 119 мілісекунд.

Програш багатопотокової версії у часі може бути обумовлений тим, що кількість даних що зчитуються не є надто великої та обробка потоків займає більше часу ніж саме зчитування з файлу.

Висновки: використання усіх ядер процесору безумовно сприяє покращенню продуктивності та часу виконання програми, але якщо використовувати багато потоків там де це недоцільно, можна спостерігати погіршення продуктивності.

Посилання на репозиторій: https://github.com/viitaliich/Concurrency