**Звіт**

З дослідженнями

До лабораторної роботи 2

Робота студента групи К-27

Лисенка Євгенія

**Мета роботи**

Метою даної лабораторної роботи було дослідження ефективності паралельного виконання стандартного алгоритму std::for\_each на різних обсягах даних. Дослідження включало порівняння швидкодії послідовного та паралельного виконання за допомогою політик виконання (std::execution::seq, std::execution::par), аналіз поведінки алгоритму при застосуванні обчислювально легкої (fast\_op) та обчислювально важкої (slow\_op) операцій. Також завданням було розробити та проаналізувати власну реалізацію паралельного алгоритму, визначити оптимальну кількість потоків K та її співвідношення з апаратною конфігурацією процесора, і оцінити вплив компіляторної оптимізації на швидкодію.

**Хід роботи**

Метою даної лабораторної роботи було дослідження ефективності паралельного виконання стандартного алгоритму std::for\_each на різних обсягах даних. Дослідження включало порівняння швидкодії послідовного та паралельного виконання за допомогою політик виконання (std::execution::seq,std::execution::par,std::execution::par\_unseq), аналіз поведінки алгоритму при застосуванні обчислювально легкої (FastOp) та обчислювально важкої (SlowOp) операцій. Також завданням було розробити та проаналізувати власну реалізацію паралельного алгоритму, визначити оптимальну кількість потоків K та її співвідношення з апаратною конфігурацією процесора, і оцінити вплив компіляторної оптимізації на швидкодію.

**Результати експериментів**

Експерименти проводилися на системі з 16 апаратними потоками. Нижче наведено зведені результати для набору даних **10,000,000 елементів**

**Таблиця 1. Час виконання (мс) для 10,000,000 елементів.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Алгоритм / Політика** | **Час для FastOp (ms)** | **Час для SlowOp (ms)** | **Прискорення для SlowOp** |
| Послідовний (seq) | ~24.5 | ~12850.1 | 1x |
| std::execution::par | ~17.2 | ~1615.4 | ~7.95x |
| **std::execution::par\_unseq** | **~18.1** | **~1598.2** | **~8.04x** |
| Власний (Best K=16) | ~19.8 | ~1630.7 | ~7.88x |

Аналіз залежності часу роботи власного алгоритму від K для операції SlowOp показав, що найкраща швидкодія досягається при **K = 16** , що дорівнює кількості апаратних потоків.

**Аналіз результатів та висновки**

Проведене дослідження показало, що ефективність паралелізму прямо залежить від обчислювальної складності задачі. Для **швидкої операції (FastOp)** виграш від розпаралелювання виявився мінімальним, оскільки **накладні витрати** на створення та синхронізацію потоків є співмірними з часом виконання самої корисної роботи. На противагу цьому, для **ресурсоємної операції (SlowOp)** переваги стали очевидними, забезпечивши майже восьмикратне прискорення, близьке до теоретичного максимуму. Це доводить, що паралелізм є виправданим лише тоді, коли тривалість обчислень значно перевищує системні витрати.

До того ж було встановлено, що оптимальна кількість потоків тісно пов'язана з архітектурою процесора. Найкращу швидкодію для власної реалізації було досягнуто при кількості потоків K, що дорівнює кількості апаратних потоків системи. При подальшому збільшенні K продуктивність деградувала через зростання **витрат на перемикання контексту**, коли операційна система витрачає більше часу на керування потоками, ніж на виконання обчислень. Важливо відзначити, що стандартна політика std::execution::par продемонструвала високу ефективність, майже не поступаючись оптимізованій власній реалізації, що підкреслює її зручність та надійність. Нарешті, критичну роль відіграла компіляторна оптимізація (-O3), яка в рази прискорила роботу програми порівняно з неоптимізованою версією.