**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, std::execution::par\_unseq), 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.

**Хід роботи**

Для досягнення поставленої мети була розроблена програма на мові C++20, яка проводила серію експериментів у режимі компіляції **Release**. Були реалізовані дві унарні операції: швидка fast\_op, що виконує просту арифметичну дію, та ресурсоємна slow\_op, яка імітує складні обчислення. Програма генерувала вектори випадкових чисел об'ємом 1,000,000 та 10,000,000 елементів. Для кожного набору даних та кожної операції вимірювався час виконання std::for\_each без політики, з політиками std::execution::seq, std::execution::par та std::execution::par\_unseq. Паралельно тестувався власний алгоритм custom\_parallel\_for\_each зі зміною кількості потоків K. Усі вимірювання часу фіксувалися для подальшого аналізу.

**Результати експериментів**

Експерименти проводилися на системі з 16 апаратними потоками. Нижче наведено зведені результати для найбільш показового 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) | 20.080 | 11900.469 | 1x |
| std::execution::par | 5.466 | 1223.026 | ~9.73x |
| std::execution::par\_unseq | 5.559 | 1187.799 | ~10.02x |
| Власний (Best K=5) | **5.253** | - | - |
| Власний (Best K=31) | - | **1054.943** | **~11.28x** |

Згідно з результатами, оптимальна кількість потоків K для власного алгоритму кардинально відрізняється для різних операцій: для FastOp найкращий час (5.2531 мс) досягнуто при **K = 5**, а для SlowOp (1054.9433 мс) — при **K = 31**.

**Аналіз результатів та висновки**

Проведене дослідження наочно демонструє, що ефективність розпаралелювання є комплексною характеристикою, яка залежить від характеру обчислювального завдання. Для ресурсоємної операції SlowOp було досягнуто значне, більш ніж 11-кратне прискорення, що підтверджує доцільність паралелізму для CPU-орієнтованих задач, де час на обчислення значно перевищує накладні витрати. Водночас оптимальна кількість потоків K виявилась нетривіальною і залежала від типу навантаження. Для інтенсивної SlowOp найкращий результат було досягнуто при K=31, що перевищує кількість апаратних потоків (16). Така поведінка є типовою для систем з технологією SMT (Hyper-Threading), де завантаження всіх логічних ядер максимізує утилізацію обчислювальних ресурсів процесора. На противагу цьому, для легкої операції FastOp, яка, ймовірно, обмежена пропускною здатністю пам'яті, оптимальним виявилося значно менше значення K=5. Це свідчить про те, що після насичення каналу пам'яті подальше збільшення кількості потоків лише створює додаткові накладні витрати. Примітно, що власна реалізація паралельного алгоритму змогла перевершити стандартні політики, що доводить переваги ручного налаштування для специфічних задач. Таким чином, лабораторна робота підтвердила, що для ефективного застосування паралелізму необхідно глибоко розуміти як саму задачу (CPU-bound vs memory-bound), так і архітектурні особливості апаратної платформи.