Звіт

Варіант 15 — transform (швидка та повільна операція)
Лазирко Ярослав Богданович К-25

Мета:

Дослідити швидкодію стандартного алгоритму std::transform з різними політиками виконання (seq, par, par_unseq) та власну реалізацію паралельного parallel_transform для різної кількості потоків K.

Хід роботи:

1. Згенеровано випадкові послідовності довжин 1000, 100000, 1000000.

2. Виконано перетворення з двома типами операцій:

    - **швидка операція:** множення на 2;

    - **повільна операція:** цикл з великою кількістю арифметичних дій.

3. Для кожного випадку виміряно час виконання:

    - без політики;

    - з політиками seq, par, par_unseq;

    - власний паралельний алгоритм при різному K.

Результати:

N = 1 000:

- Fast: std::transform = 1.8e-6 сек, seq = 1.1e-6, par = 4.29e-5, par_unseq = 2.6e-6
  Власний паралельний алгоритм: найкраще K = 1, час = 1.6e-6, апаратні потоки = 12

- Slow: std::transform = 0.00028 сек, par = 0.000129, par_unseq = 8.59e-5
  Найкраще K = 4, час = 0.000218, апаратні потоки = 12

N = 100 000:

- Fast: std::transform = 0.000129, par_unseq = 0.000062, найкраще K = 1
- Slow: std::transform = 0.0281, par_unseq = 0.00269, найкраще K = 23

N = 1 000 000:

- Fast: std::transform = 0.00129, par_unseq = 0.000198, найкраще K = 8
- Slow: std::transform = 0.280, par_unseq = 0.0264, найкраще K = 12

Висновки:

1. При збільшенні кількості потоків до значення, близького до кількості апаратних потоків процесора, спостерігається мінімальний час виконання.
2. Подальше збільшення K призводить до зростання накладних витрат на створення потоків.
3. Алгоритм std::transform(execution::par_unseq) показує подібну або трохи кращу продуктивність порівняно з власною реалізацією при великому N.
4. Для коротких послідовностей переваги паралелізації майже непомітні.
5. Режим Release істотно зменшує час виконання (у кілька разів порівняно з Debug).