Звіт з програмного проєкту #2. Варіант 6. Скалярний добуток

Бондар Ігор, К-25

1 Дослідження швидкодії бібліотечного алгоритму за відсутності політики

Було отримано наступну залежність середнього часу виконання `inner_product` (кількість ітерацій, за якою обчислювалось середнє значення - 100) від рівня оптимізації компілятора та розміру набору даних, результати в секундах:

	-O0	**-O3**
10^6	00.002907146	00.001685511
10^7	00.029529218	00.010245602
10^8	00.286894775	00.116723436

Можна побачити, що порівняно з відсутністю оптимізації компілятора, рівень оптимізації -O3 дає прискорення швидкості у 1.7-2.7 рази залежно від розміру набору даних. Збільшення розміру набору даних в 10 разів супроводжувалось збільшенням середенього часу дії також приблизно в 10 разів при кожному з рівнів оптимізації.

2 Дослідження швидкодії бібліотечного алгоритму з різними політиками

Тестові дані є середнім значенням для 100 ітерацій, результати в секундах. Замість `inner_product` було використано `transform_reduce`.

При -O0 отримано такі результати:

	seq	**par**	**unseq**	**unseq_par**
10^6	00.004791194	00.001446464	00.004285078	00.001088002
10^7	00.049858826	00.010177948	00.043263378	00.009737234
10^8	00.461493398	00.100224756	00.463815651	00.094641129

При -O3 отримано такі результати:

	seq	**par**	**unseq**	**unseq_par**
10^6	00.001244226	00.001080762	00.001291248	00.001151432
10^7	00.011561653	00.010565542	00.008871664	00.010063270
10^8	00.092731851	00.099454970	00.090902973	00.098251966

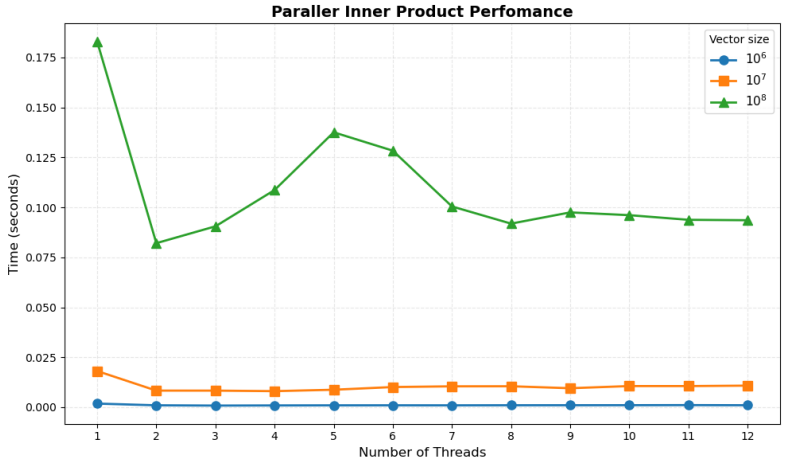
При переході від -O0 до -O3 спостерігається прискорення в 5-50 разів залежно від політики (найбільше прискорення відбувається з `unseq`). При максимальній оптимізації алгоритм найшвидше виконується при `par` та `unseq` залежно від розміру тестового набору даних.

3 Дослідження швидкодії власного паралельного алгоритму за умови використання різної кількості потоків

Тестові дані є середнім значенням для 100 ітерацій, результати в секундах.

Threads	10^6 **Time (s)**	10^7 **Time (s)**	10^8 **Time (s)**
1	0.001831738	0.018110411	0.182832350
2	0.000975452	0.008323423	0.082071916
3	0.000835001	0.008316808	0.090501939
4	0.000908644	0.008053734	0.108652922
5	0.000955740	0.008760594	0.137530988
6	0.000971425	0.010115881	0.128333309
7	0.000957492	0.010465109	0.100446176
8	0.000996461	0.010501224	0.091850681
9	0.001007595	0.009521683	0.097469641
10	0.001023551	0.010600083	0.096079841
11	0.001067615	0.010596632	0.093758749
12	0.001002216	0.010818058	0.093548843

Для датасету розміром 10^6 алгоритм найшвидше виконується при 2–3 потоках, для 10^7 — 2–4, для 10^8 — при 2 потоках. Це значення корелює з кількістю фізичних ядер процесора. Оптимальна продуктивність досягається при кількості потоків, що не перевищує або приблизно дорівнює кількості фізичних ядер процесора. Для великих задач (10^8) спостерігається більш виражена деградація продуктивності при збільшенні кількості потоків, що пояснюється інтенсивнішою конкуренцією за пам'ять та кеш процесора.



При збільшенні кількості потоків до 2-3 час виконання зменшується, після точки оптимиму на 2-4 потоках, йде збільшення часу виконання відповідно до збільшення кількості потоків.