Politechnika Krakowska
Wydział Informatyki i Telekomunikacji

Programowanie Równoległe i Rozproszone

Sprawozdanie z projektu

Zespół:
Karol Hamielec
Maryna Lukachyk
Krzysztof Dulęba

Temat:
Znajdowanie k-tego największego elementu ze zbioru liczb (bez sortowania). Zaimplementuj w środowiskach OpenMP, MPI oraz środowisku hybrydowym MPI+OpenMP. Porównaj wyniki.

Wprowadzenie
Celem projektu było zaimplementowanie algorytmu znajdowania k-tego największego elementu ze zbioru liczb, bez sortowania. Algorytm miał być zaimplementowany w trzech różnych środowiskach równoległych: OpenMP, MPI oraz środowisku hybrydowym MPI+OpenMP, a następnie należało porównać uzyskane wyniki w każdym ze środowisk pod kątem czasu wykonywania.

Etap 1. Implementacja bazowego programu sekwencyjnego w języku C.
Program implementuje algorytm QuickSelect, który znajduje k-ty największy element ze zbioru liczb bez sortowania. Działanie tego algorytmu oparte jest na zasadzie „dziel i zwyciężaj”.

Sposób działania algorytmu:

1. Wybieramy pivot z tablicy.
2. Porównujemy pivot z pozostałymi elementami tablicy i dzielimy ją na dwie podtablice - jedną zawierającą elementy mniejsze od pivota, drugą zawierającą elementy większe od pivota.
3. Jeśli wartość k jest mniejsza od pozycji pivotu, to wywołujemy algorytm rekurencyjnie na pierwszej podtablicy, w przeciwnym razie na drugiej.
4. Algorytm kończy się, gdy pivot ma pozycję k.

Etap 2. Implementacja programu współbieżnego w środowisku OpenMP.

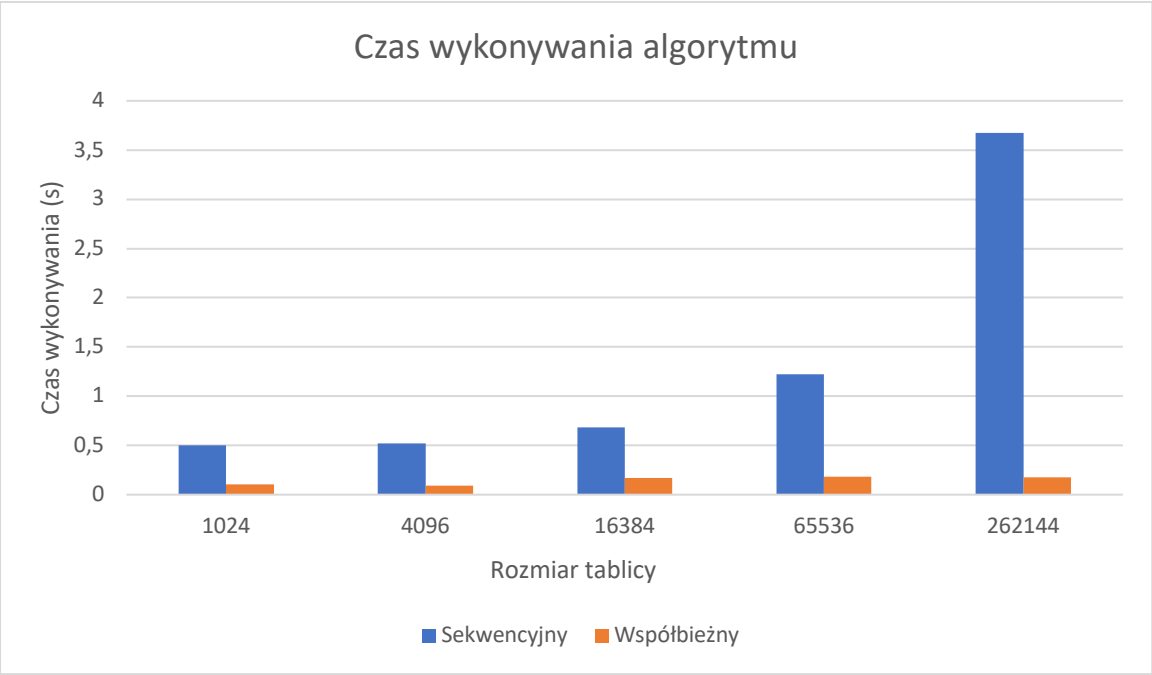
Program korzysta z wielowątkowości, wykorzystując bibliotekę OpenMP, do podziału tablicy na mniejsze podtablice, które są sortowane niezależnie w każdym wątku. Podział tablicy na podtablice odbywa się za pomocą funkcji partition(), która dzieli tablicę na dwie części względem wybranej wartości pivota (punktu odniesienia). Główna funkcja programu parallel_quickselect_no_alloc() przeprowadza wstępne rozdzielenie tablicy na mniejsze podtablice i wywołuje funkcję partition() dla każdej z tych podtablic w osobnym wątku. Następnie łączy wyniki i ustala kolejną wartość pivota, którego używa do ponownego podziału na podtablice. Jeśli znaleziony k-ty element jest mniejszy niż liczba elementów w danej podtablicy, algorytm rekurencyjnie wywołuje partition() na mniejszej podtablicy. Jeśli znaleziony k-ty element jest większy, to wywołuje partition() na większej podtablicy.

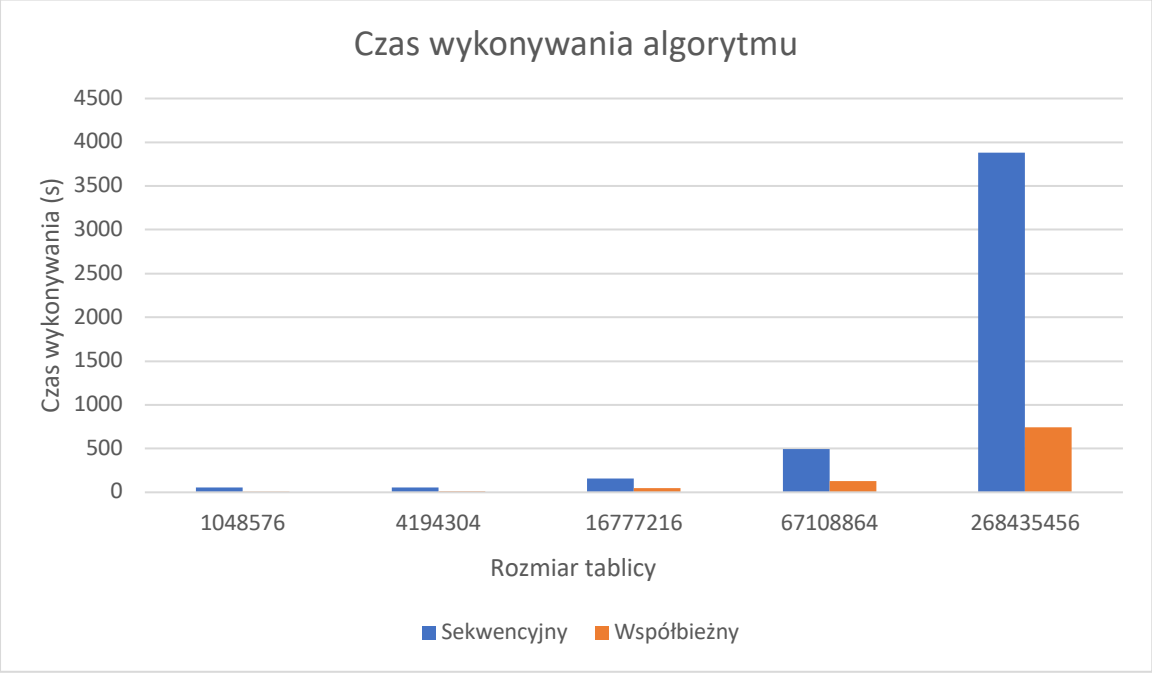
Etap 3. Testy algorytmu sekwencyjnego oraz współbieżnego (OpenMP) i porównanie wyników

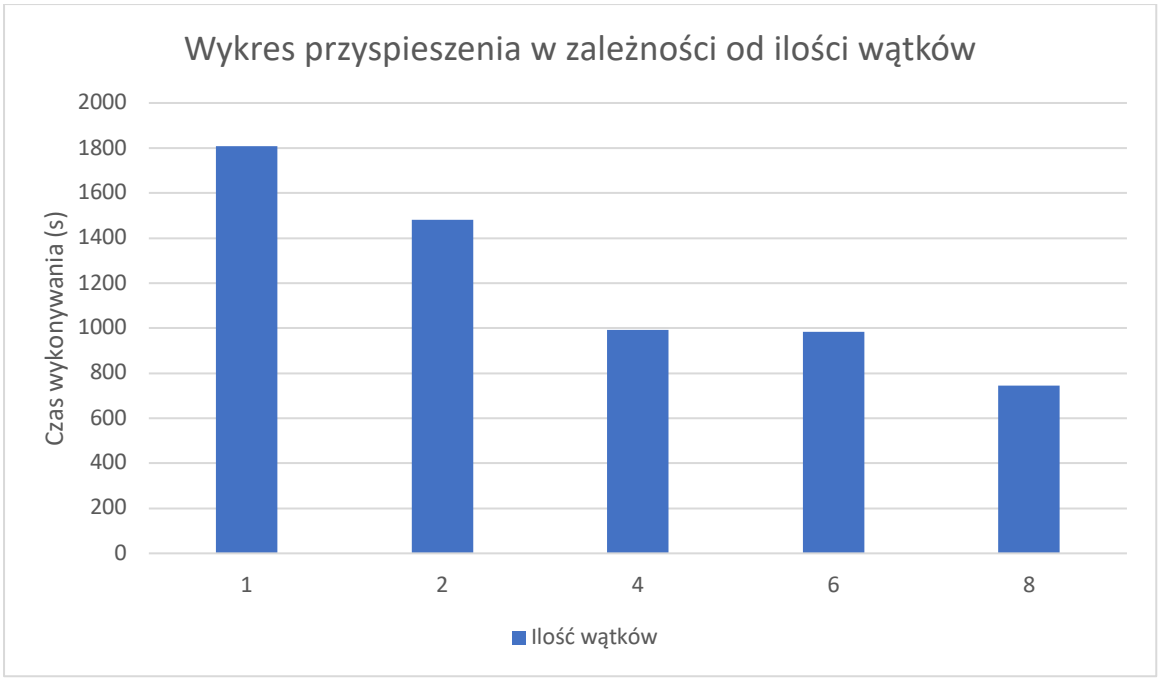
Program napisany w języku C z implementacją algorytmu zawiera funkcję, która służy do zapisania w pliku CSV wyników pomiarów czasowych dla różnych trybów działania programu. Utworzono również skrypty pomocnicze w języku Python – preprocess.py oraz runner.py. Skrypty te służą do uruchomienia algorytmu dla różnych wielkości tablic i ilości wątków, a następnie przetworzenia otrzymanych wyników czasowych i wygenerowania plików wynikowych CSV. W celu uzyskania rzetelnych wyników, ostateczne wyniki czasowe są średnią z wyników otrzymanych wskutek 5 uruchomień programu. Na podstawie ostatecznych wyników czasowych, utworzone zostały utworzone diagramy prezentujące uzyskane wyniki badania.

Platforma testowa użyta do przeprowadzenia testów:
System operacyjny: Linux
Procesor: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
RAM: 16GB
Maksymalna liczba wątków programu uruchamianego współbieżnie: 8

Na podstawie uzyskanych wyników utworzono poniższe diagramy:






Wykres przyspieszenia w zależności od ilości wątków
Czas wykonywania (s)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
1
2
4
6
8
Ilość wątków

Wnioski

Na wybranej platformie testowej, niezależnie od rozmiaru tablicy, zauważono wielokrotnie korzystniejsze czasy wykonania algorytmu przy zastosowaniu równoległego środowiska OpenMP, niż w przypadku wersji sekwencyjnej algorytmu. Badając przyspieszenie algorytmu w zależności od liczby wątków, stwierdzono, że największe i porównywalne skoki nastąpiły po zwiększeniu liczby wątków z 1 do 2 oraz z 2 do 4. Zwiększenie liczby wątków z 4 do 6 oraz z 6 do 8 również przyniosło poprawę czasów wykonania, ale w mniejszym stopniu.