Politechnika Krakowska im. Tadeusza Kościuszki
Wydział Informatyki i Telekomunikacji

Zrównanie algorytmu znajdowania największego elementu ze zbioru liczb

10 stycznia 2022

Tomasz Grzesik, Piotr Ksel

1 Wprowadzenie

Celem projektu jest przedstawienie różnic w czasie weryfikacji, która liczba w zbiorze jest największa. Zostanie omówiony model z wymianą komunikatów oraz model wirtualnej pamięci wspólnej pamięci wspólnej. Na końcu zaimplementowany został model hybrydowy.

1.1 Treść zadania

Znajdowanie k-tego największego elementu ze zbioru liczb.

1.2 Dane wejściowe

Każde z wersji algorytmu posiada dane wejściowe, które są losowane generatorem liczb pseudolosowych. Poniżej funkcja odpowiedzialna za generowanie liczb.

```
for (int i(0); i < silk; i++)
    nums.push_back(rand() % 10000);
```

2 Wersja w MPI

Model z wymianą komunikatów (Message Passing Interface) charakteryzuje się podziałem problemu na podproblemy, które są opracowywane przez odrębne procesy. Podproblemami są podzbiory wejściowego zbioru danych. Zbiór wejściowy został podzielony na niezależne fragmenty, dla których niezależne się oblicza k-tą największą wartość. W procesie zerowym zostały wygenerowane dane wejściowe i zapisane do tablicy. W modelu z wymianą komunikatów dane wymieniane pomiędzy procesami przesyłane są za pomocą komunikatów. Implementacja programu w MPI wykorzystuje przesyłanie komunikatów typu jeden do jeden.

```
MPI_Status status;
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &th);
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
double startParallel, stopParallel;
startParallel = MPI_Wtime();
for (int i(0); i < silk; i++)
    nums.push_back(rand() % 10000);

MPI_Send(&silk, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
MPI_Recv(&k, 1, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &status);
```

3 Wersja w OpenMP

Model pamięci wspólnej algorytmu do wyznaczenia k-tej największej wartości w standardzie OpenMP (Open Multi-Processing). Standard ten wykorzystuje pracę na wielu wątkach oraz pamięć współdzieloną. Kod algorytmu został napisany w języku C++. Implementacja algorytmu odróżnia się od MPI brakiem przesyłania komunikatów, ponieważ wykorzystana została pamięć wspólna. Kod jest tożsamy z wersją sekwencyjną, jednakże pętla algorytmu została zrównoleglona. Zrównoleglony kod algorytmu przedstawia rysunek poniżej.

```
    double startParallel, stopParallel;
    startParallel = omp_get_wtime();
#pragma omp parallel for reduction(+:sum,quantity) num_threads(th)
    for (int i(0); i < silk; i++)
        nums.push_back(rand() % 1000);

    cout << "Array:";
    cout << k << " Largest2:" << KthLargest2(nums, k) << endl;
    stopParallel = omp_get_wtime();
```

4 Wersja hybrydowa

Implementacja hybrydowa łączy oba modele programowania równoległego: model z wymianą komunikatów i model pamięci wspólnej. Kod charakteryzuje się dwoma pozmiomami zrównoleglenia. Kod programu początkowo rozdziela zadania na poszczególne procesy za pomocą funkcji MPI, natomiast w obrębie sprawdzenia warunku uruchamiane są dyrektywy OpenMP.

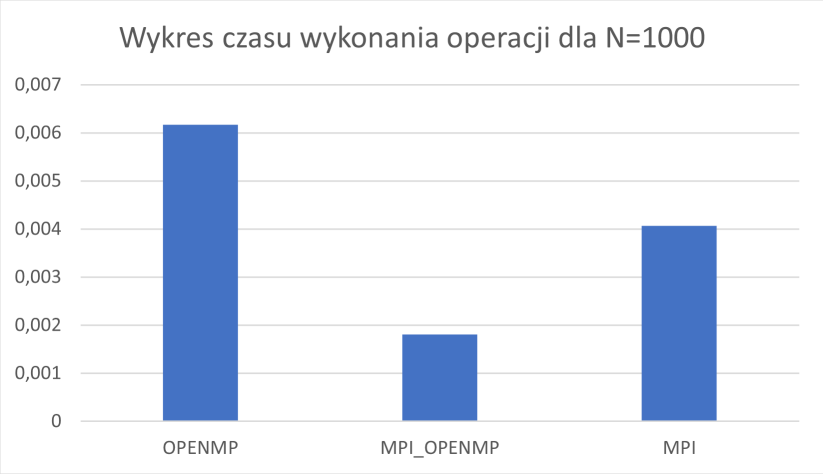
```
MPI_Status status;
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &th);
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
double startParallel, stopParallel;
startParallel = MPI_Wtime();
#pragma omp parallel for num_threads(th)
for (int i(0); i < silk; i++)
    nums.push_back(rand() % 1000);

MPI_Send(&silk, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
MPI_Recv(&k, 1, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &status);
cout << "Array:";
cout << k << " Largest2:" << KthLargest2(nums, k) << endl;
```

5 Czasy operacji

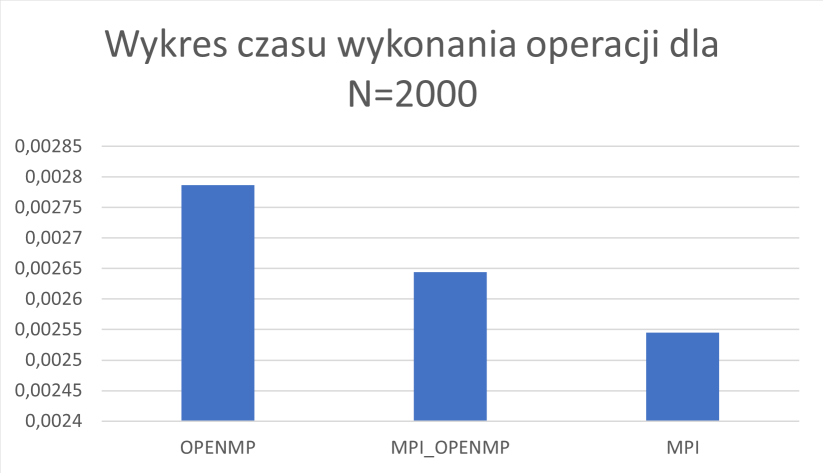
Czasy wykonania operacji zostały sprawdzone dla 1 tysiąca elementów w tablicy, 2 tysięcy elementów, ORAZ 5 tysięcy elementów tablicy.

5.1 N=1000



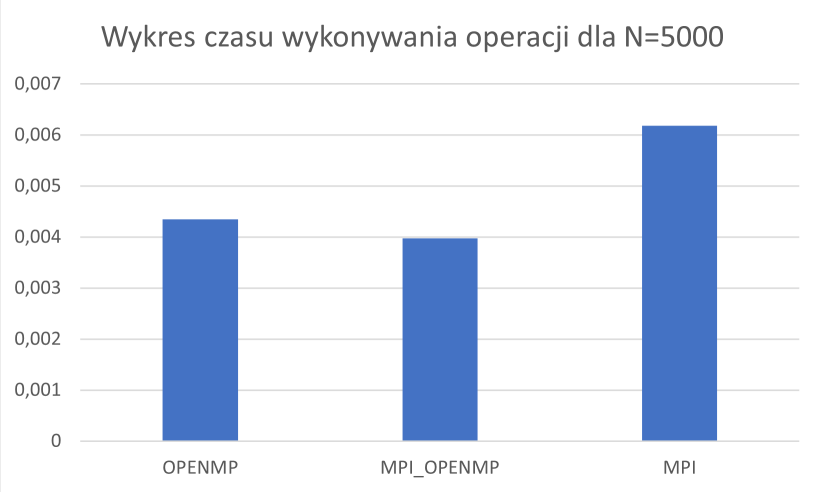
Dla 1 tysiąca elementów w tablicy optymalny jest algorytm MPI.

5.2 N=2000



Dla 2 tysiący elementów w tablicy optymalny jest ponownie algorytm MPI.

5.3 N=2000



Dla 2 tysiący elementów w tablicy optymalny jest ponownie algorytm MPI.

6 Podsumowanie

Dla testowanych wartości N najszybsze działanie wykazywał algorytm MPI.