Politechnika Krakowska im. Tadeusza Kościuszki
Wydział Informatyki i Telekomunikacji

Zrównanie algorytmu znajdowania największego elementu ze zbioru liczb

8 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.

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 wykonywane były niezależne obliczenia średniej arytmetycznej przez poszczególne procesy. W procesie zerowym zostały wygenerowane dane wejściowe i zapisane do tablicy. Każdy proces obliczał dla fragmentu tablicy średnią arytmetyczną liczb o zdefiniowanym zakresie, wykorzystując pętlę for. W modelu z wymianą komunikatów dane wymieniane pomiędzy procesami przesyłane są za pomocą komunikatów. Implementacja programu w MPI wykorzystuje przesył komunikatów typu jeden do jeden. Dzięki zastosowaniu funkcji MPI Send() oraz MPI Recv() referencja do wejściowego zbioru danych oraz obliczane sumy cząstkowe były przekazywane pomiędzy procesami co umożliwiło obliczenie średniej arytmetycznej liczb w tablicy.

```
MPI_Init(&argc, &argv);

unsigned int vSize = 1000000;
if (argc == 2) {
    vSize = atoi(argv[1]);
}

unsigned int fillMode = FILL_MODE_NOT_RANDOM;
if (argc == 3) {
    if (strcmp("-random", argv[3])) {
        fillMode = FILL_MODE_RANDOM;
    }
}

int procCount, procIndex;


MPI_Comm_size(MPI_COMM_WORLD, &procCount);
MPI_Comm_rank(MPI_COMM_WORLD, &procIndex);



double startParallel, stopParallel;
startParallel = MPI_Wtime();


double* dataVector;
double procMax;
```

3 Wersja w OpenMP

Model pamięci wspólnej algorytmu obliczającego średnią liczb został wykonany 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.

```
#pragma omp parallel for shared(a)
    for (i = 0; i < n; i++) {
#pragma omp critical
        {
            if (a[i] > mx)
                mx = a[i];
        }
    }
    int smx = -1;
    for (i = 0; i < n; i++)
        smx = a[i] > smx ? a[i] : smx;
    printf("\nSerial max = %d\tParallel max = %d\n", smx, mx);
    timeDiff = omp_get_wtime() - time;
    printf("Time: %f", timeDiff);
```

4 Wersja hybrydowa

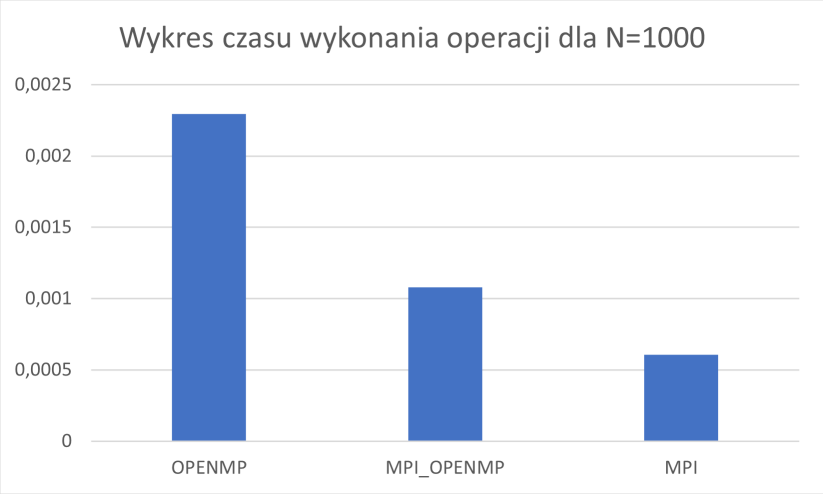
Implementacja hybrydowa łączy oba modele programowania równoległego: model z wymianą komunikatów i model pamiędzi wspólnej. Kod charakteryzuje się dwoma pozmiomami zrównoleglenia. Kod programu początkowo rozdziela zadania na poszczególne procesy za pomocą funkcji MPI, natomiast w obrębie sprawdzenia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

```
if (procIndex == 0) {
    printf("Array size per process is %u\n", vSize);
}


procMax = dataVector[0];
for (unsigned int i = 0; i < vSize; i++) {
    #pragma omp critical
    if (dataVector[i] > procMax) {
        procMax = dataVector[i];
    }
}

MPI_Reduce(&procMax, &calculatedMax, 1, MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);

if (procIndex == 0) {
    if (fillMode == FILL_MODE_NOT_RANDOM) {
        if (calculatedMax == procCount - 1) {
            printf("Max parallel = %f\n", calculatedMax);
        }
        else {
            printf("No Success Calculation Maximum! You must recalculate this!");
            return 1;
        }
    }
}
```

5 Czasy operacji

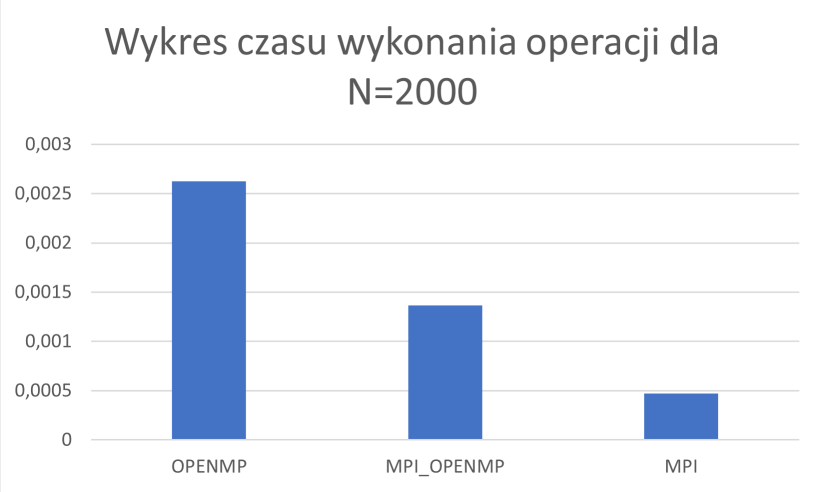
Czasy wykonania operacji zostały sprawdzone dla 1 tysiąca elementów w tablicy oraz 2 tysięcy elementów tablicy.

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.

6 Podsumowanie

Dla testowanych wartości N najszybsze działanie wykazywał algorytm MPI.