POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

Programowanie równoległe i rozproszone
Projekt część III – użycie MPI – Message Passing Interface
Zadanie BK27 - ,,Słownik równoległy”

Prowadzący: dr inż. Bartłomiej Kubica
Wykonał: Szymon Nogieć
Nr indeksu: 234446

1. Implementacja

Główną pętlą programu jest pętla for iterująca poprzez wszystkie elementy przeznaczonej do wyszukania. Na początku programu pobierany jest czas metodą std::chrono::high_resolution_clock::now() oraz ponownie po wykonaniu programu.

Program główny został zainicjalizowany wywołaniem MPI_Init (int argc, char **argv[]) a zakończenie bloku wywołaniem MPI_Finalize().

Główną blok równoległy programu jest rozpoczynany wywołaniem metod:

```
MPI_Comm_rank(MPI_COMM_WORLD, &mynum);
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
```

Metody te pozwalają na pobranie liczby wartości rank danego procesu oraz liczby wszystkich procesów odpowiednio do zmiennych mynum i nprocs.

W kolejnym kroku główne pętle programu przebudowano w ten sposób, że proces o wartości rank 0 jest odpowiedzialny za kolejno:

a. Pobranie stringa z wektora słów do wyszukania,

b. Pobranie nproc-1 kolejnych stringów z wektora do wyszkania i rozesłanie ich do pozostałych procesów,

c. Wyszukanie występowania danego słowa w swojej kopii słownika,

d. Oczekiwanie i pobieranie wyników działań od nproc-1 procesów oraz umieszczenie ich w docelowej tablicy,

e. Zwrócenie wyniku

2. Pozostałe nproc-1 procesów po wejściu w główną pętle:

a. Oczekiwało na otrzymanie od procesu o identyfikatorze 0 stringa do wyszukania,

b. Wykonywało oczekiwaną operację,

c. Wysyłało wyniki obliczeń (zmienna bool – czy znaleziono) do procesu o rank 0 i oczekiwało na kolejną porcję danych.

Do komunikacji pomiędzy procesami wykorzystano komunikację asynchroniczną z synchronizacją wzajemną, a więc metody MPI_Send (void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm) oraz MPI_Recv (void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status). Dodatkowo, aby nie przesyłać dodatkowo rozmiaru tablicy znaków słowa do wyszukania dodano wywołanie MPI_Probe w celu sprawdzenia statusu przesyłanych informacji do n-tego procesu i następnie wywołanie MPI_Get_count(&status, MPI_CHAR, &l); w celu pobrania rozmiaru tablicy i zaalokowania odpowiedniego bufora na ciąg znaków.

2. Wyniki obliczeń

Dla porównania obliczeń wykorzystano tekst o rozmiarze około 2MB. Obliczenia przeprowadzono na procesorze 4-rdzeniowym intel i5-4590 w systemie operacyjnym linux dystrybucja Ubuntu 18.04. Wykorzystano kompilator g++ w wersji 7.3.0. Do projektu wykorzystano CMakeFile w którym określono linkowanie biblioteki i nagłówków MPI.

Program uruchamiano za pomocą

Liczba wątków\operacja	Kodowanie [s] Dekodowanie [s]
1	6.58353

2	9.68216
4	12.3544

Całość wyników programu:

➜ cmake-build-debug git:(feature/mpi) ✗ mpirun -n 1 ./mpi_prir
----OPENMP PARALLEL SEARCH----
find mpi done
MPI time took 6.58353 with 1 processors.
➜ cmake-build-debug git:(feature/mpi) ✗ mpirun -n 2 ./mpi_prir
----OPENMP PARALLEL SEARCH----
find mpi done
MPI time took find mpi done
9.68216 with 2 processors.
➜ cmake-build-debug git:(feature/mpi) ✗ mpirun -n 4 ./mpi_prir
----OPENMP PARALLEL SEARCH----
find mpi donefind mpi done
find mpi done
find mpi done

MPI time took 12.3544 with 4 processors.

Do testów wykorzystano takie same rozmiary danych wejściowych jak w przypadku poprzedniej iteracji.

3. Wnioski

Zdecydowanie wyniki są niezadowalające. Niewątpliwie wynika to ze sposobu implementacji słownika i potrzeby wymiany danych pomiędzy procesami. Dane przesyłane między procesami są małe (odpowiednio tablica MPI_CHAR oraz MPI_CXXBOOL), w związku z czym zysk wnikający z wykorzystania wielu procesów zostaje przewyższony przez czas potrzebny na synchronizacje powyższych.

Warto zauważyć, że jeśli powyższa komunikacja zostałaby wykorzystana w rozdzieleniu dużo bardziej wymagających pod względem obliczeniowym (a więc także czasowym) zadań, zysk zauważalny (wzrost dążący do liniowego). Dodatkowo należy zaznaczyć, że obliczenia wykonywano na maszynie lokalnej z pamięcią współdzieloną.

Do raportu załączono pliki źródłowe programu.