Klasyfikowanie dokumentów - program typu Manager - Workers

Systemy równoległe i rozproszone

Łukasz Wajda, Mariusz Biegański

8 maja 2023

1 Opis projektu

W internecie jest wile dokumentów, aby znaleźć właściwy dokument za pomocą wyszukiwarki wymyślono algorytm zliczający ilość słów w dokumentach i zapisujący to w postaci wektora. Projekt wykorzystuje rozmaite pliki tekstowe łatwe do przetwarzania takie jak .txt, .odt, .css, .html, .xml. Korzystając z pliku zawierającego listę słow, tzw. słownik tworzymy wektor zawierający ilość wystąpnień danego słowa dla każdego pliku. Na koniec wyniki zapisujemy do pliku.

2 Opis budowy programu

Program został napisany z wykorzystaniem biblioteki MPI (Message Passing Interface). Składa się z trzech funkcji: *main*, *manager* i *worker* oraz enum *Tags* do reprezentacji stałych wartości.

3 Opis działania programu

3.1 Funkcja *main*

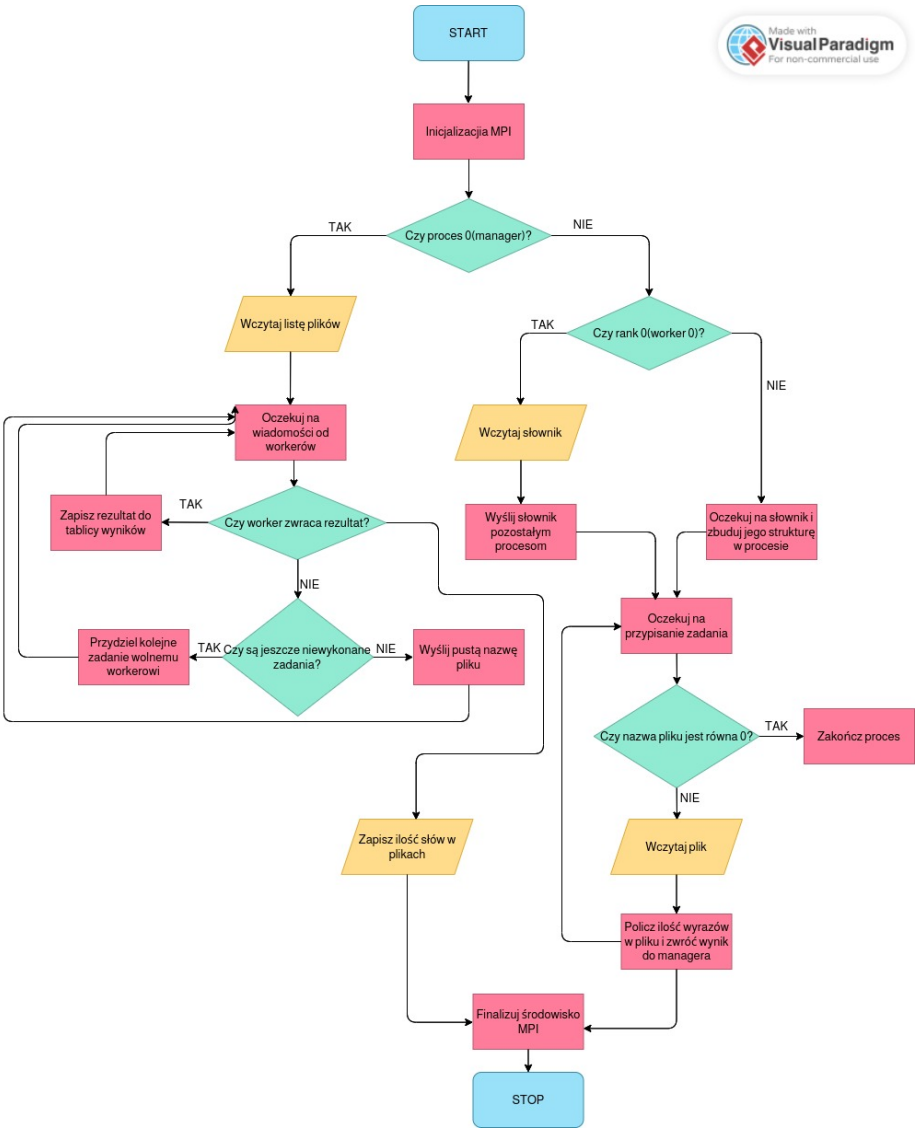
Inicjalizujemy MPI i uzyskujemy informacje o liczbie procesów i identyfikatorze bieżącego procesu. Następnie tworzony jest komunikator worker_communicator, który będzie używany do komunikacji między menadżerem a workerami. Jeśli identyfikator procesu wynosi 0 (czyli jest to menadżer), przypisuje się proces do komunikatora o kolorze undefined za pomocą funkcji *MPI_ Comm_ split*, po to aby nie komunikował się tak jak worker. Następnie wywołuje funkcję manager przekazując jej argumenty argc, argv oraz liczbę procesów num_processes. W przeciwnym razie, jeśli identyfikator procesu nie jest równy 0 (czyli jest to worker), funkcja main wywołuje *MPI_ Comm_ split* dla komunikatora *MPI_ COMM_ WORLD*, tworząc nowy komunikator worker_communicator dla workerów o kolorze 0. Następnie wywołuje funkcję worker przekazując jej argumenty argc, argv oraz komunikator worker_communicator. Na końcu funkcja main wywołuje *MPI_ Finalize* w celu zamknięcia środowiska MPI i zwraca wartość 0.

3.2 Funkcja *manager*

Na początku zostaje wywołana asynchroniczna funkca czekająca na dowolny proces, który prześle informację o ilości słów w słowniku otagowaną *DICTIONARY_ SIZE_ MESSAGE*. Podczas czekania manager przeszukuje podany folder i zapisuje nazwy plików do wektora napisów. Po wczytaniu listy program jest blokowany i czeka na zakończenie funkcji asynchronicznej. Następnie alokowane jest potrzebne miejsce na wyniki zwrócone przez funkcje *worker* z liczbą wystąpień słow w danym pliku. Po zaalokowaniu pamięci wykonuje się pętla while do czasu zakończenia pracy wszystkich funkcji *worker*. W pętli manger czeka na wiadomość od workera. Jeśli jest ona otagowana *FILE_ VECTOR_ MESSAGE* to wynik zostaje dodany do zmiennej przechowującej wektory wystąpień dla każdego pliku. Dalej w pętli sprawdza się czy liczba przypisanych plików jest mniejsza od całkowitej liczby plików, jeśli tak to przesyłana zostaje nazwa pliku do funkcji *worker*, w przeciwnym przypadku zostaje przesłany komunikat do *worker* o końcu pracy. Po zakończeniu pętli wyniki zostają zapisane do pliku *result.txt*.

3.3 Funkcja *worker*

Funkcja pobiera swój identyfikator procesu, następnie wysyła pusty komunikat informujący o gotowości do pracy. Jeśli worker ma identyfikator równy zero to wczytuje słownik z pliku i koduje go tablicy charów oddzielając słowa znakiem '|'. Następnie rozmiar słownika zostaje rozgłoszony funkcją *MPI_ Bcast*. Pozostałe workery używają otrzymanej długości słownika do zaalokowania pamięci na słownik. Następnie zostaje rozgłoszony zakodowany słownik do pozostałych workerów. Workery o indeksie różnym od zera dekodują otrzymany słownik i zapisują do wcześniej zaalokowanej zmiennej. Po rozgłoszeniu słownika worker o indeksie zero przesyła do managera ilość słów w słowniku. Na koniec wykonuję się nieskończona pętla while, w której worker sprawdza czy jest dostępna wiadomość otagowana *SOURCE_ FILE_ MESSAGE* za pomocą funkcji *MPI_ Probe*. Jeśli tak to pobierana jest długość nazwy pliku do przetworzenia. W przypadku gdy długość nazwy wynosi zero pętla zostaje przerwana i kończy się funkcja. Po uzyskaniu długości nazwy zostaje pobrana nazwa pliku, a następnie zapisana do lokalnej zmiennej. Plik zostaje wczytany, stworzony zostaje wektor wystąpień, który jest przesyłany do funkcji *manager*. Po zakończeniu pętli zostają zwolnie zasoby.



Rysunek 1: Schemat blokowy

4 Obsługa programu

Program kompilujemy poleceniem *make*, uruchamiamy poleceniem *make run*, a czyścimy projekt do stanu początkowego poleceniem *make clean*.
Słownik znajduje się w pliku *data/checkedWords.txt* i można go modyfikować. Dodatkowo można dodawać i usuwać pliki tekstowe do sprawdzenia w folderze data. Wynik można wyświetlić w terminalu za pomocą komendy *cat result.txt*,

należy pamiętać by wyświetlić wynik po uruchomieniu programu i przed przywróceniem projektu do stanu początkowego. Przykładowy wynik znajduje się w pliku example.txt i można go wyświetlić w terminalu za pomocą komendy *cat example.txt*.