*Programowanie równoległe i rozproszone*

Równoległe/rozproszone wyszukiwanie informacji (wyszukiwarka internetowa) [JB20]

# Wstęp

Kod został podzielony na odpowiednie funkcje, które realizują konkretne powierzone im zadanie. Całość jest okomentowana, tak aby osoba trzecia mogła szybko wywnioskować co dany fragment kodu oznacza.

Do wyszukiwania został wykorzystany algorytm knutha-morrisa-pratta.

# Przykładowy całościowy przebieg programu

Po podaniu ścieżki oraz słowa klucza do wyszukania całkowity wynik (część sekwencyjna bez optymalizacji) operacji to;

Keyword to search (max 100 characeters): znajdz mnie

wyniki wyszukiwania:

========================

/home/marcin/Education/files/essays20

634579

637285

========================

/home/marcin/Education/files/essays11

========================

/home/marcin/Education/files/essays25

2719639

2720800

========================

/home/marcin/Education/files/essays6

========================

/home/marcin/Education/files/essays21

========================

/home/marcin/Education/files/essays22

========================

/home/marcin/Education/files/essays1

4

3160603

========================

/home/marcin/Education/files/essays10

3160594

3160606

========================

/home/marcin/Education/files/essays7

========================

/home/marcin/Education/files/essays23

========================

/home/marcin/Education/files/essays12

========================

/home/marcin/Education/files/essays18

========================

/home/marcin/Education/files/essays9

========================

/home/marcin/Education/files/essays13

========================

/home/marcin/Education/files/essays4

========================

/home/marcin/Education/files/essays15

2239970

2241919

========================

/home/marcin/Education/files/essays24

========================

/home/marcin/Education/files/essays2

========================

/home/marcin/Education/files/essays19

========================

/home/marcin/Education/files/essays17

========================

/home/marcin/Education/files/test

1

========================

/home/marcin/Education/files/essays3

========================

/home/marcin/Education/files/essays14

========================

/home/marcin/Education/files/essays8

========================

/home/marcin/Education/files/essays5

1474806

2327180

========================

/home/marcin/Education/files/essays16

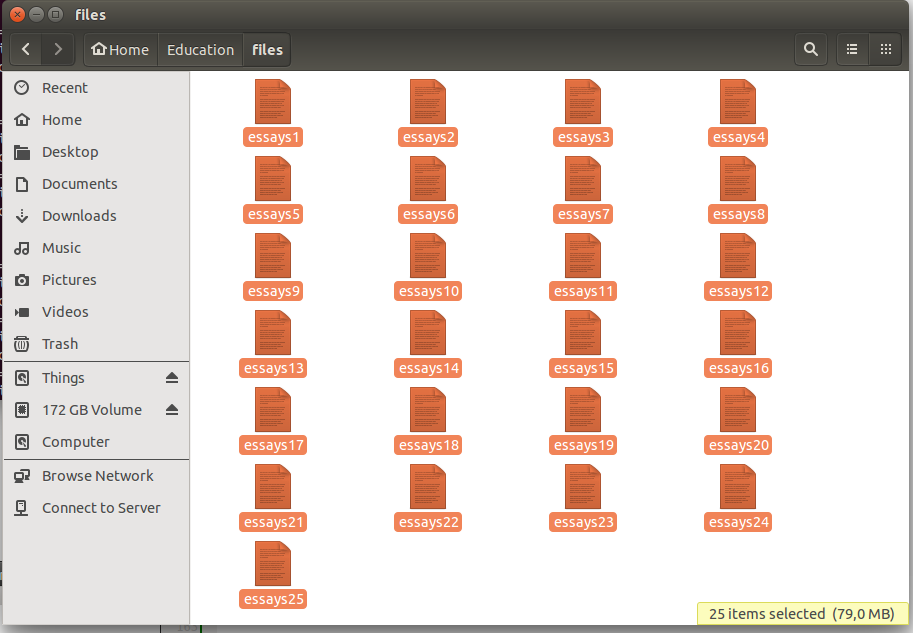
Czas: 0.666403

Suma wystąpień:13

W dalszych częściach dokumentu listowanie nazwy plików oraz numery indeksu wystąpienia wzorcu w tekście zostaną pominięte – będziemy analizować głównie wynik końcowy, czyli czas wykonania oraz porównanie wydajności po zrównolegleniu kodu.

# Środowisko testowe

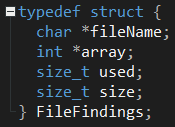
Linux Ubuntu 15  
Intel Core i5-2430M 2,40 GHz  
12 GB RAM, Dysk SSD.

Liczba plików która była przeszukiwana; 25  
łączny rozmiar plików; 79MB  
liczba znaków w 1 pliku: Około 3 000 000.

# Najtrudniejsze problemy

Sam język C nie jest językiem prostym (jeśli ktoś na codzień pracuje z językami typu C#). Największym problemem okazało się wyświetlanie wyników na ekranie, które dobrze działało w części sekwencyjnej, a przysporzyło sporo problemów po zrównolegleniu – Tekst po prostu nie wyświetlał się po kolei (wynikowe numery indeksów początku wystąpienia wyświetlały się pod losowymi nazwami plików), a po wymuszeniu kolejności (#pragma omp ordered) wydajność drastycznie spadła, zrównując się niemal z częscią sekwencyjną.

Została wprowadzona struktura która przechowuje nazwę plików oraz tablicę indeksów występień w tym pliku. Następnie kod został zmieniony, tak aby dynamiczna tablica tych struktur była uzupełniana w kolejnych iteracjach pętli a końcowe wyświetlenie wyniku ma miejsce na samym końcu programu, tuż przed zwalnianiem pamięci. To pomogło przyśpieszyć wykonanie programu w części sekwencyjnej i zrównolegnionej, oraz wygrać walkę o kolejność wyświetlania wyników.



# Sposób pomiaru czasu

Część sekwencyjna:

clock\_t start = clock();

//główna pętla która odczytuje i przeszukuje pliki

clock\_t end = clock();

float seconds = (float)(end - start) / CLOCKS\_PER\_SEC;

OMP:

double start\_time = omp\_get\_wtime();

//główna pętla która odczytuje i przeszukuje pliki

double time = omp\_get\_wtime() - start\_time;

# Zrównoleglenie przy użyciu OMP

Zrównoleglenie zostało zrealizowane poprzez dodanie dyrektywy;  
**#pragma omp parallel for default(none) shared(filesCount, fileFindings, T, keyWord)**

Dzięki której wskazuję która pętla ma zostać zrównolegniona, oraz jakie zmienne powinny być współdzielone przez kolejne wątki.

Do ustawiania liczby wątków została użyta dyrektywa;

**omp\_set\_dynamic(0);** //Wyłączenie dynamicznych “teams”  
**#pragma omp parallel for num\_threads(4) default(none) shared(filesCount, fileFindings, T, keyWord)**

Wersji z ustaloną liczbą wątków nie znajdziemy w kodzie, było to używane tymczasowo na potrzeby testów.

Kompilacja odbywała się używając polecenia;

**gcc main.c -fopenmp -O2 -march=native -o lookup**

# Pomiary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| słowo klucz | Suma wystąpień | Sekwencja | 2 wątki | 4 wątki | 8 wątków | 16 wątków | Dynamic teams |
| znajdz mnie | 13 | 0,35367 | 0,231107 | 0,21739 | 0,160037 | 0,158098 | 0,157369 |
| testing | 300 | 0,428407 | 0,252467 | 0,176931 | 0,215913 | 0,157631 | 0,211862 |
| probably | 5700 | 0,37946 | 0,240129 | 0,187718 | 0,160264 | 0,13971 | 0,187175 |
| I am | 8100 | 0,36031 | 0,191979 | 0,183538 | 0,170625 | 0,164519 | 0,156766 |
| as | 332400 | 0,407677 | 0,254654 | 0,208911 | 0,197708 | 0,18671 | 0,174931 |

Wykres (słowo „as zostało pominięte, gdyż ilość wystąpień zaburzała widoczność reszty wykresu):

# Podsumowanie

OMP ponad dwukrotnie przyśpieszyło działanie programu (w zależności od liczby wątków), który moim zdaniem i w części sekwencyjnej jest piekielnie szybki. W przyszłości jeśli przyjdzie pracować mi z czymś gdzie wydajność jest kluczowa – na pewno sięgnę po OMP z językiem niskopoziomowym myśląc już wcześniej o odpowiednim zaprojektowaniu pętli (Teraz przekopałem prawie cały kody).