Sprawozdanie z Projektu Pierwszego

Laboratorium Przetwarzania Równoległego

Piotr Tylczyński
L7 / 141331
Środa, 11:45
piotr.tylczynski@student.put.poznan.pl

Zuzanna Rękawek
L7 / 141304
Środa, 11:45
zuzanna.rekawek@student.put.poznan.pl

Oddane: 29.04.2021
Deadline: 29.04.2021

Wersja 1

Contents

1 Motywacja

Celem niniejszego projketu jest stworzenie efektywnego programu wyszukującego liczby pierwsze w zadanym przedziale. W tym celu wykorzystamy porogramowanie równoległe. Pozwoli to na efektywniejsze wykorzystanie zasobów komputerowych jakimi dysponujemy. W wyniku otrzymamy program mogący wykorzystywać do 100% mocy obliczeniowej procesora komputera, na którym zostanie uruchomiony. Pozwoli to nam na znaczącą redukcję czasu wykonania programu względem standardowej wersji sekwencyjnej programu.

W rozwiązaniu wykorzystamy algorytm Sita Erastotenesa (SE). Jest to algorytm pozwalający na osiągnięcie znaczącego przyspieszenia, po jego zrównolegleniu. Dodatkowo rozważymy wykorzystanie i zrównoleglanie algorytmu Pełnego Przeglądu List Liczb (PPLL).

2 Specyfikacja platformy uruchomieniowej

Procesor Intel Core i5-9300H

Procesorów Fizycznych 4

Procesorów Logicznych 8

Pamięć Cache 8 MB Intel® Smart Cache

System Operacyjny Windows 10 Pro 20H2

IDE Visual Studio 2019

Oprogramowanie Testujące 5t4iori

3 Problemy i zjawiska występujące w programach równoległych

3.1 Problemy Poprawnościowe

3.1.1 Zjawisko wyścigu

Jest to zjawisko polegające na ubieganiu się o dostęp do pojedynczego zasobu przez wiele wątków. Jest to warunek konieczny wystąpienia w.w. zjawiska. Jeżeli nie zastosujemy, żadnego systemu zarządzania dostępem okaże się, że wynik wykonania programu nie będzie deterministyczny. Będzie to spowodowane przez używanie jednego zasobu przez wiele wątków. Będą one potencjalnie zapisywały do zasobu równocześnie. Spowoduje to, że wartość zapisana będzie w najlepszym przypadku wartością zapisaną przez ostatni wątek. W gorszym przypadku powstanie losowa kombinacja wartości zapisywanych przez poszczególne wątki. Kolejość informacji zapisanych w zasobie będzie zależała od czasu dostępu, stąd nazwa wyścig.

Oczywiście występowanie zjawiska wyścigu nie jest porządana. Może ona doprowadzić do powstania złych wyników. W najgorszym przypadu będą one wypadkową, lub permutacją poprawnych wyników.

Rozwiązaniem problemu wyścigu jest wprowadzenie ograniczeń w dostępie do współdzielonego zasobu. W najprostszy sposób można to osiągnąć blokując zasób w czasie jego wykorzystywania przez wątek. Wtedy pozbędziemy się problemu wielodostępu i nigdy nie dopuścimy do spełnienia warunku koniecznego. Narzędziem, które może okazać się przydatne mogą być monitory, lub semafory. W naszym projekcie zastosujemy odpowiednie klauzule, które zapobiegną opisywanemu zjawisku. Będą to klauzule synchronizujące i dzielącę prace między wątkami, które zostały szczegółowo opisane poniżej, jaki i w opisach samych algorytmów.

3.2 Problemy Efektywnościowe

3.2.1 Problem False Sharing

Polega na unieważnieniu potencjalnie nie współdzielonych danych w pmięci podręcznej procesora. Dzieje się tak, ponieważ leżą one na tej samej lini adresowej procesora. Z tego powodu jeżeli procesor wykonuje zapis do jednej z komórek pamięci to może się okazać, że narusza inną zmienną leżącą na tej samej lini pamięci. W takim wypadku, zmienna ta zostanie uznana za *"brudną"* i będzie wymagała ponownego pobrania z pamięci.

Zjawisko False Sharingu prowadzi do znaczengo spadku efektywności programu. W skrajnych przypadkach może okazać się, że program jest znacząco wolniejszy od swojej wersji sekwencyjnej. Jest to bezpośrednie następstwo wielokrotnego i niepotrzebnego unieważniania liń pamięci. Co sprawia, że wymagany jest dodatkowy narzut czasowy związnay z transferem danych pomiędzy poszczególnymi poziomami pamięci komputera.

4 Zastosowane Algorytmy

4.1 Opis teoretyczny

4.1.1 Sito Erastotenesa

W swojej najprostszej formie SE jest alogrytmem, który przyjmuje na swoje wejście wektor liczb naturalnych, uporządkowanych rosnąco z krokiem jeden - w dalszych częściach teog dokumentu nazywanych Wektorem Liczb Uporządkowanych - ()WLU). Zastosowanie WLU pozwala na dokonanie pewnej optymalizacji. Polega ona na sprawdzaniu tylko liczb znajdujących się przed połową takiego wektora. Optymalizacja taka jest możliwa, ponieważ WLU jest rosnąco uporządkowany, więc wiemy, że wszystkie liczby złozone w drugiej połowie są, wielokrotnością, liczb znajdujących się w pierwszej połowie. Biorąc pod uwagę sposób działania algorytmu, który wykreśla z WLU wszystkie wielokrotności

liczb, mamy pewność, że po przejrzeniu wszystkich liczb z pierwszej połowy, wyeliminowaliśmy wszystkie pozostałe z drugiej połowy.

Sam sposób działania SE jest prosty. Dla każdej znalezionej liczby w WLU wykreśl z WLU wszystkie jej wielokrotności - w ten sposób pozbywamy się liczb złożonych. Następnie przejdź do kolejnej liczby w WLU i powtórz poprzedni krok. Algorytm ten działa o ile przeglądamy liczby z zakresu 1 do N (N dowolna liczba całkowita). Jeżeli pierwszą liczbą w WLU nie jest 1 to należy stworzyć sztuczny iterator, który będzie przechodził przez dodatkową tablicę zawierającą wszystkie liczby pierwsze (TLP). Kolejną optymalizację jaką można dokonać jest ograniczenie wielkości tablicy liczb peirwszych. Maksymalna wymagana liczba pierwsza to pierwiastek kwadratowy z ostatniej liczby wchodzącej w skład WLU. Tą własność można łatwo udowodnić, ponieważ największy dzielnik dowolnej liczby naturalnej nie może być większy niż pierwiastek kwadratowy z niej samej.

4.1.2 Przegląd Listy Liczb

Jest to jden z najprostszych algorytmów wyszukiwania liczb pierwszych. W swojej sekwencyjnej wersji polega na pełnym przejrzeniu WLU i znalezieniu w nim liczb pierwszych. Jest to algorytm mniej skuteczny niż SE, jednak szybszy w implementacji.

4.1.3 Określanie pierwszości liczyb

Działa na zasadzie iterowania się po wszystkich liczbach z zakresu 2 do wartości liczby. Jednocześnie sprawdzamy, czy któraś z liczb nie jest dzielnikiem sprawdzanej liczby. Jeżeli tak to wiemy, że dana liczba jesst liczbą złożoną. Po zakończeniu iterowania i nie znalezieniu dzielnika, możemy stwierdzić, że sprawdzana liczba jest pierwsza. Optymalizacja algorytmu polega na sprawdzaniu liczby tylko i wyłącznie nie większych niż pierwiastek kwadratowy z testowanej liczby. Uzasadnienie tego faktu można znaleźć w algorytmie SE.

4.2 Realizacja praktyczna

4.2.1 Wykorzystane klauzule OMP

`omp_get_max_threads()` podaje maksymalną dostępną w systemie liczbę procesorów logicznych

`omp_get_wtime()` zwraca czas pracy wszystkich wątków

`#pragma omp parallel` rozpoczyna obszar wykonania Równoległego

`#pragma omp for` pozwala na zrównoleglanie pętli dzieląc wykonywaną przez nią pracę pomiędzy dostępne wątli

`omp_get_thread_num()` zwraca numer aktualnie wykonywanego wątku

`omp_set_num_threads()` ustawia maksymalną ilość wątków jakiej może używać program

Glossary

PPLL Pełny Przegląd Listy Liczb. 2

SE Sito Erastotenesa. 2, 3, 4

TLP Tablica Liczb Pierwszch. 4

WLU Wektor Liczb Uporządkowanych. 3, 4