**Jędrzej Kuczyński**

**sprawozdanie z projektu 2 OMP**

**Wstęp**

Przedmiot: Laboratorium z obliczeń wielkiej skali

Autor: Jędrzej Kuczyński, numer indeksu 414728 (WB UAM)

Terminy: 3.06.2020 rzeczywisty;

Wersja pierwsza

Opis: Analiza przetwarzania równoległego dwóch algorytmów znajdowania liczb pierwszych.

Adres email: jk.kuczynski@gmail.com

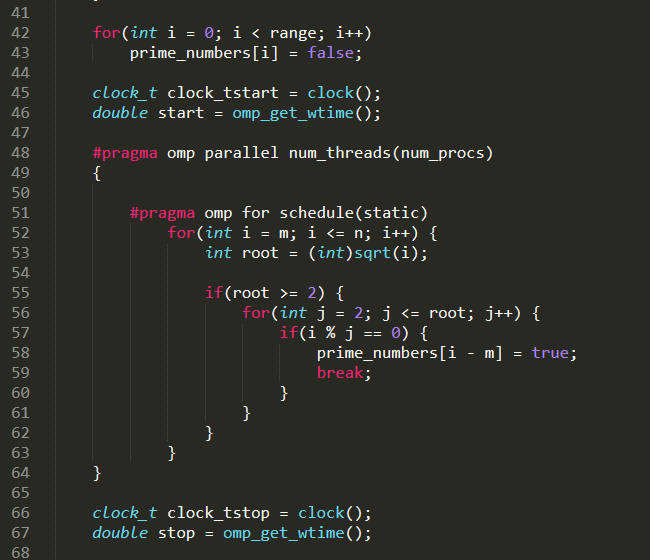
**1. Wykorzystywany system obliczeniowy**

Procesor: Intel Core i5-4690 3.50GHz; liczba rdzeni: 4; liczba wątków: 4. 6MB Intel Smart Cache

System operacyjny i kompilator: Ubuntu 18.04.4 LTS; gcc 7.5.0

**2. Warianty kodu i ich omówienie**

**a) poprzez dzielenie:**



*Rysunek 1. Algorytm wyszukujący liczby pierwsze poprzez dzielenie - wariant static (wariant dynamic wygląda analogicznie).*

Algorytm wyszukujący liczby pierwsze poprzez dzielenie nie korzysta z samych liczb pierwszych jako potencjalnych dzielników, jednakże ogranicza ich liczbę do pierwiastka z badanej liczby. Pojedyncze zadanie przydzielane wątkowi składa się ze zbioru liczb, którego wielkość jest zależna od rozmiaru instancji i liczby procesów uczestniczących w obliczeniach.

Sposób przydziału pracy:

a) statyczny blokowy - zakres iteracji pętli dzielony jest na podzakresy o jednakowym rozmiarze (po równo, dodatkowe iteracje wątkom o mniejszym id), każdy wątek dostaje jeden zakres; blokowy - kolejne iteracje dla tego samego wątku. Bazuje na zakresie pętli wyznaczonym w czasie przetwarzania.

b) dynamiczny - wątki zwracają się do modułu szeregującego w celu otrzymania pierwszej iteracji, a po wykonaniu jej następnych. Algorytm działał na domyślnej wielkości przydziału równej 1.

Wyścig, false sharing i synchronizacja:

Wyścig to zjawisko, kiedy jeden proces operuje na nieaktualnych danych ze względu na to, iż inny równolegle, w tym samym czasie je zmodyfikował. W tej wersji algorytmu wyścig nie występuje, ponieważ każdy z procesów uczestniczących w obliczeniach działa na własnej części tablicy.

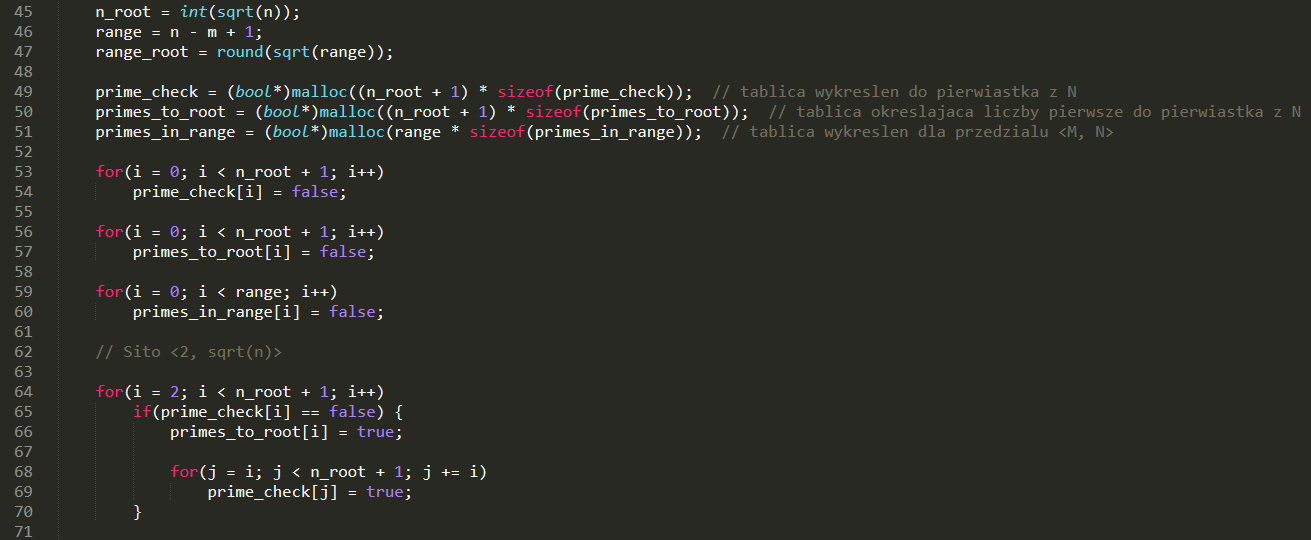
False sharing to sytuacja powodująca dodatkowe narzuty czasowe wynikające z unieważnień kopii danych poprzez zapisy przez różne procesory różnych słów ulokowanych logicznie w tym samym obszarze linii pamięci podręcznej - efektem zapisu jest unieważnienie wszystkich nieaktualnych kopii zapisanej linii w innych procesorach. Taka sytuacja wymaga wstrzymania dostępu do danych unieważnionych i zaktualizowanie ich do najnowszej wersji. False sharing występuje w tej wersji algorytmu, ponieważ procesy działają na wspólnej tablicy służącej wyznaczaniu liczb pierwszych.

Synchronizacja następuje po wyjściu z pętli *#for* oraz podczas aktualizowania unieważnionych tablic.

Przewidywany przebieg przetwarzania:

Powyższy algorytm powinien działać lepiej, pod względem przyspieszenia, efektywności i kosztu zrównoleglenia, ze sposobem przydziału pracy dynamicznym aniżeli statycznym blokowym. Spowodowane jest to tym, że mniejsze liczby należy sprawdzić pod kątem mniejszej liczby jak i mniejszych ogólnie dzielników. W przydziale pracy statycznym blokowym pierwsze iteracje, a więc liczby mniejsze, otrzymają procesy o mniejszym id i ukończą swe obliczenia o wiele wcześniej niż reszta procesów - przez resztę działania algorytmu nie będą przeprowadzać żadnych obliczeń. Taka sytuacja nie występuje w dynamicznym przydziale pracy, ponieważ w chwili zakończenia obliczeń przez proces otrzyma on kolejne zadanie - algorytm będzie działał szybciej.

**b) sito domenowe:**

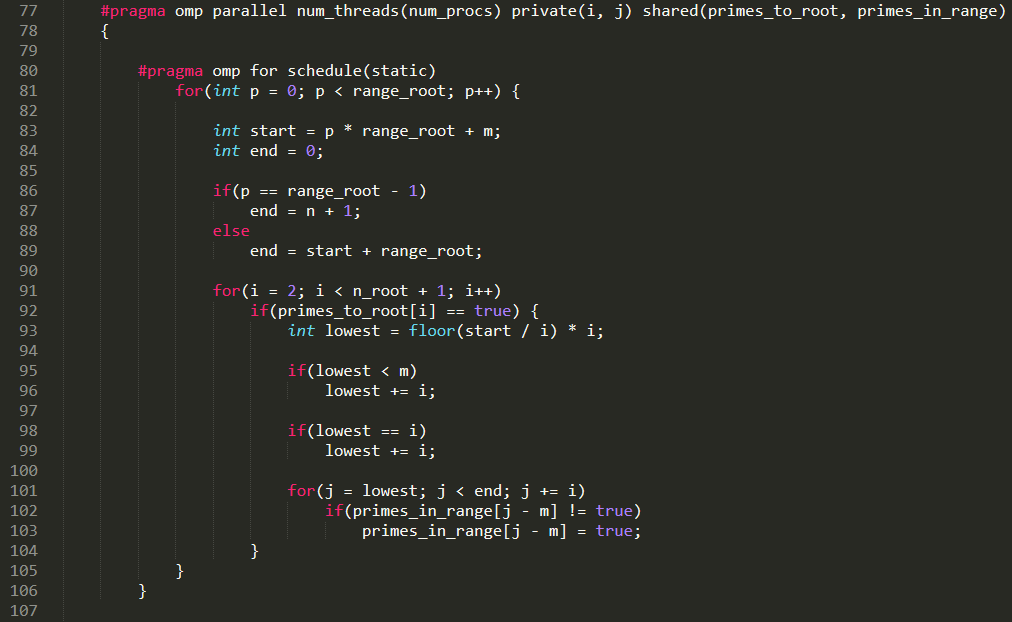


*Rysunek 2. Część kodu wyszukująca liczby pierwsze do pierwiastka z zakresu.*

***n\_root - pierwiastek kwadratowy z zakresu***

***range - długość zakresu***

***range\_root - pierwiastek kwadratowy z długości zakresu. Na tyle podtablic będzie dzielona tablica.***

******

*Rysunek 3. Główna pętla algorytmu sita domenowego - wariant static (wariant dynamic wygląda analogicznie).*

W ramach każdej pętli określany jest początek i koniec podtablicy, którą dany proces w aktualnej iteracji będzie przetwarzał. Następnie w ramach każdej wcześniej znalezionej liczby pierwszej znajdowana jest jej najmniejsza wielokrotność znajdująca się w przetwarzanym przedziale. Znalezione minimum jak i kolejne wielokrotności aktualnej liczby pierwszej są wykreślane.

Pojedyncze zadanie podlegające przydziałowi to znalezienie w danej części tablicy wszystkich wielokrotności, wszystkich liczb pierwszych znalezionych do pierwiastka kwadratowego z zakresu. Wielkość zbioru zależy od długości badanego zakresu.

Sposób przydziału pracy:

a) statyczny blokowy - zakres iteracji pętli dzielony jest na podzakresy o jednakowym rozmiarze (po równo, dodatkowe iteracje wątkom o mniejszym id), każdy wątek dostaje jeden zakres; blokowy - kolejne iteracje dla tego samego wątku. Bazuje na zakresie pętli wyznaczonym w czasie przetwarzania.

b) statyczny cykliczny - deterministycznie jednakowa liczba grup iteracji dla wszystkich wątków, dodatkowe grupy dla wątków o mniejszym id. Kolejne grupy iteracji (o danej wielkości) przydzielane kolejnym wątkom:

Przykład: 2 wątki, 9 iteracji. przedział = 2 (0, 2, 4, 6, 8, 10, 12, 14, 16):

wątek 0 - grupy (0, 2), (8, 10), 16; sumarycznie 5 iteracji

wątek 1 - grupy (4, 6), (12, 14); sumarycznie 4 iteracje

b) dynamiczny - wątki zwracają się do modułu szeregującego w celu otrzymania pierwszego zakresu iteracji, a po wykonaniu go następnych.

Wyścig, false sharing i synchronizacja:

Wyścig nie występuje, jako że każdy proces w danej iteracje otrzymuje część nienachodzącej na siebie głównej tablicy.

False sharing występuje, ponieważ procesy działają na podtablicach o sporym rozmiarze będących częścią głównej tablicy.

Synchronizacja następuje po wyjściu z pętli *#for* oraz podczas aktualizowania unieważnionych tablic.

Przewidywany przebieg przetwarzania:

W porównaniu do algorytmu sita funkcyjnego, algorytm domenowy będzie wydajniejszym algorytmem, ponieważ w każdej iteracji dany fragment tablicy jest ładowany do pamięci podręcznej raz i w ramach niego znajdowane są wszystkie liczby pierwsze. W porównaniu do strategii z dzieleniem, tego typu algorytm również będzie wydajniejszy, przede wszystkim ze względu na różnice w podstawowych operacjach wykonywanych na przetwarzanych liczbach - znajdowanie wielokrotności poprzez dodawanie jest o wiele szybsze i wydajniejsze niż dzielenie.