**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; 3.09.2020 termin oddania

Wersja pierwsza

Opis: Analiza przetwarzania równoległego dwóch algorytmów znajdowania liczb pierwszych.

Adres email: jk.kuczynski@gmail.com

**1. Wykorzystywany system obliczeniowy**

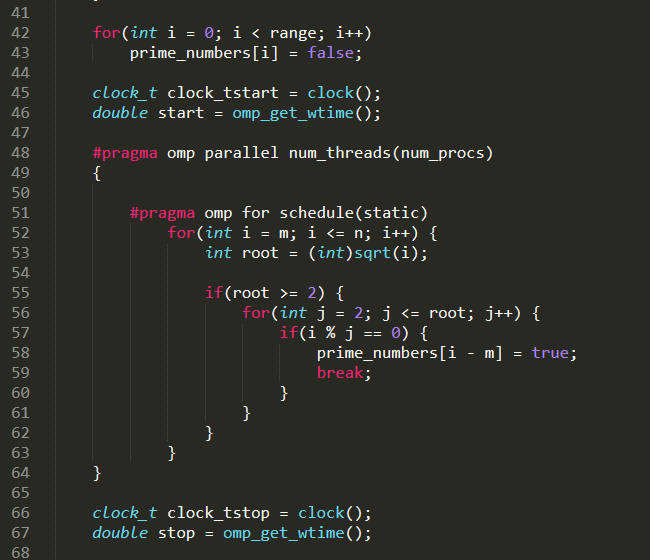
Procesor: Intel Core i5-4690 3.50GHz; liczba rdzeni: 4; liczba wątków: 4. 6MB Intel Smart Cache

System operacyjny i kompilator: Ubuntu 18.04.4 LTS; gcc 7.5.0

Pamięć RAM: 8GB

**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.

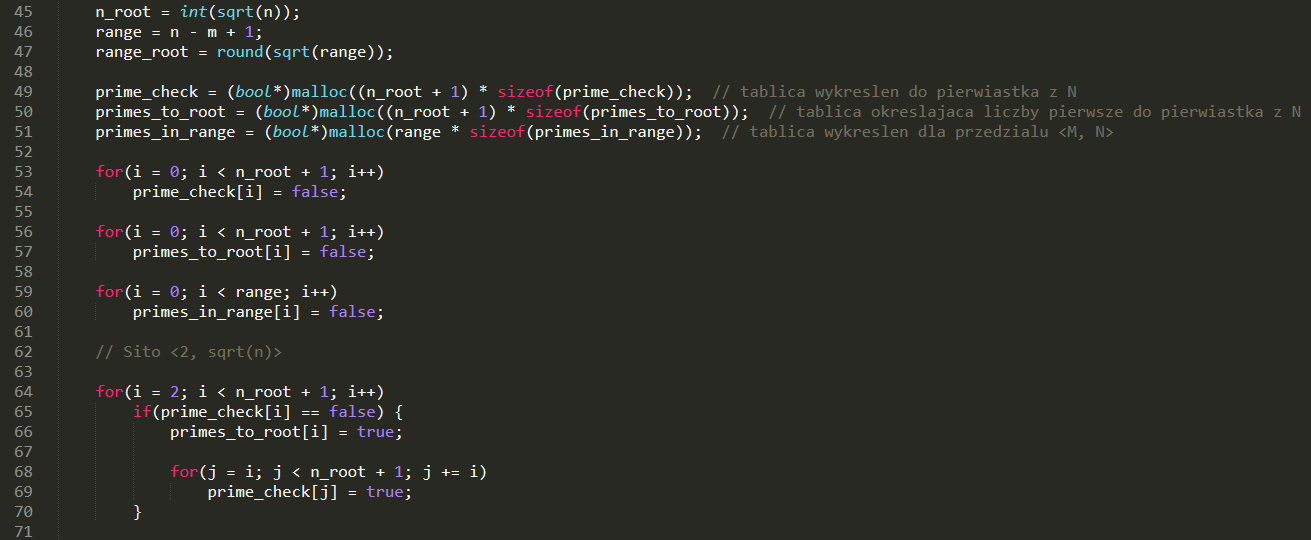
False sharing to sytuacja powodująca dodatkowe narzuty czasowe wynikające z unieważnień kopii danych poprzez zapisy przez różne procesory różnych słów ulokowanych logicznie w tym samym obszarze linii pamięci podręcznej - efektem zapisu jest unieważnienie wszystkich nieaktualnych kopii zapisanej linii 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:**

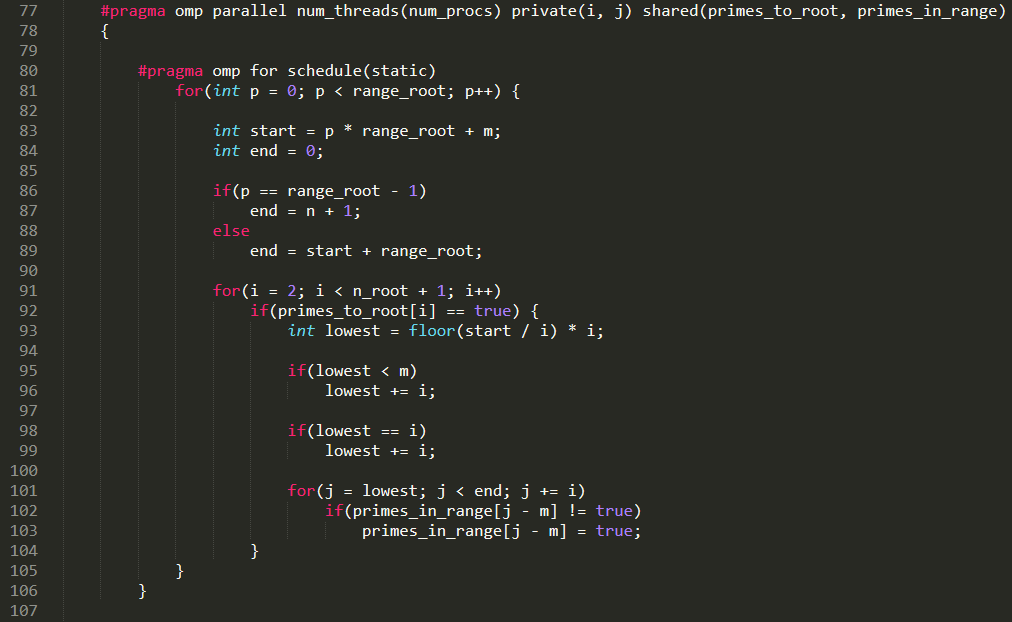


*Rysunek 2. Część kodu wyszukująca liczby pierwsze do pierwiastka z zakresu.*

***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 iteracji otrzymuje część nienachodzącej na siebie głównej tablicy.

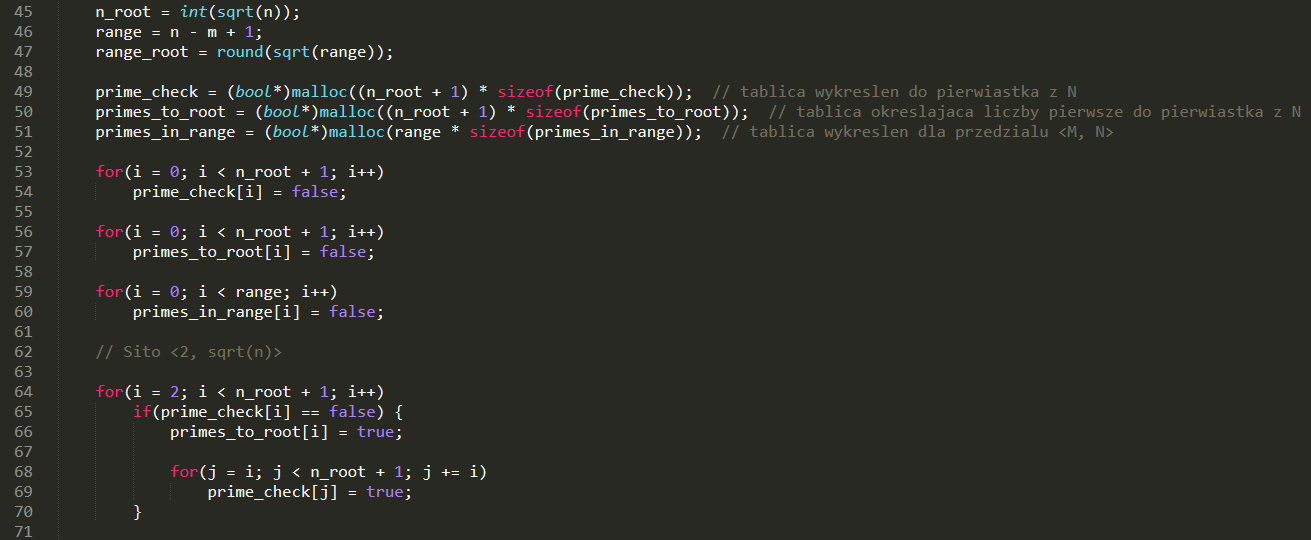
False sharing występuje, ponieważ procesy działają na podtablicach o sporym rozmiarze będących częścią głównej tablicy.

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

Przewidywany przebieg przetwarzania:

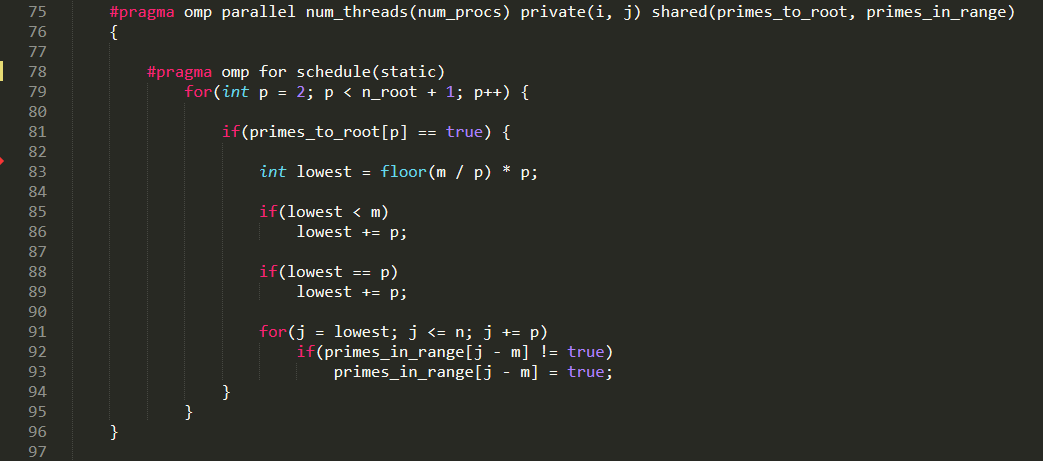
W porównaniu do algorytmu sita funkcyjnego, algorytm domenowy będzie wydajniejszym algorytmem, ponieważ w każdej iteracji dany fragment tablicy jest ładowany do pamięci podręcznej raz i w ramach niego znajdowane są wszystkie liczby pierwsze. W porównaniu do strategii z dzieleniem, tego typu algorytm 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.

**c) sito funkcyjne:**

****

*Rysunek 4. Część kodu wyszukująca liczby pierwsze do pierwiastka z zakresu.*

Jest to ten sam fragment kodu, co w algorytmie sita domenowego.



*Rysunek 5. Główna pętla algorytmu sita funkcyjnego - wariant static (wariant dynamic wygląda analogicznie).*

W algorytmie sita funkcyjnego procesy otrzymują zbiór liczb pierwszych, a następnie wykreślają w całej tablicy wszystkie liczby będące ich wielokrotnościami. Poza tym, główna pętla algorytmu działa analogicznie do poprzedniego kodu.

Pojedyncze zadanie podlegające przydziałowy to znalezienie w całej tablicy wszystkich wielokrotności z danego podzbioru liczb pierwszych. Wielkość całego zbioru liczb pierwszych oraz blok liczb podlegający przydziałowy zależy od pierwiastka kwadratowego z górnego zakresu badanego przedziału.

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 występuje w tej wersji algorytmu, ponieważ może dojść do sytuacji, w której dwa procesy będą przetwarzały takie liczby pierwsze, które mają wspólne wielokrotności. Jednakże to zjawisko nie prowadzi do otrzymania złych wyników, dopóki któryś z procesów wykreśli wielokrotność zmieniając wartość boolowską na przeciwną (false --> true) - ponowna taka zmiana nadal doprowadzi do prawidłowych wyników, a zmiana na oryginalną wartość jest niemożliwa, ze względu na warunek sprawdzający w kodzie.

False sharing występuje, ponieważ procesy działają na wspólnej tablicy i ciągle zmieniają jej wartości.

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

Przewidywany przebieg przetwarzania:

Ta wersja algorytmu będzie działała wolniej i mniej wydajnie niż sito domenowe - w tej wersji algorytmu procesy kopiują fragmenty tablicy do pamięci podręcznej, wykreślają z nich wielokrotności przyznanej liczby pierwszej, a następnie powtarzają ten proces dla kolejnej przyznanej liczby pierwszej. To wszystko powoduje duże nakłady czasowe związane z ciągłym odwoływaniem się do pamięci oraz odczytem/zapisem z/do niej. Natomiast w porównaniu do strategii z dzieleniem, tego typu algorytm 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.

**3. Wyniki**

**a) poprzez dzielenie:**

Algorytm był uruchamiany i testowany na wielkościach instancji, które zapewniały czas obliczeń poniżej 1 minuty:  
 <2, 60 000 000> (2, MAX), <2, 30 000 000> (2, MAX/2), <30 000 000, 60 000 000> (MAX/2, MAX).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Sekwencyjny** | **Static 2P** | **Dynamic 2P** | **Static 4P** | **Dynamic 4P** |
| **Czas pracy procesorów** | 53,51497 | 53,39491 | 53,25413 | 54,22289 | 56,06154 |
| **Upływ czasu rzeczywistego** | 53,54109 | 33,54488 | 26,62803 | 18,16539 | 14,04532 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,596103 | 2,010704 | 2,947423 | 3,812023 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,798052 | 1,005352 | 0,736856 | 0,953006 |
| **Prędkość przetwarzania** | 1 120 635 | 1 788 648 | 2 253 264 | 3 302 984 | 4 271 885 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 40,82548 | 0,007263 | 101,5044 | 0,852604 |
| **Koszt zrównoleglenia** | BRAK | 13,69486 | 0,001934 | 18,43868 | 0,119751 |

***Tabela 1. Tabela przedstawiająca miary efektywnego przetwarzania dla przedziału liczb <2, 60 000 000> (2P - algorytm był uruchamiany na 2 procesach itp.; parametry przydziału static/dynamic domyślne).***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Sekwencyjny** | **Static 2P** | **Dynamic 2P** | **Static 4P** | **Dynamic 4P** |
| **Czas pracy procesorów** | 19,45305 | 19,52668 | 20,01151 | 20,21892 | 21,07541 |
| **Upływ czasu rzeczywistego** | 19,45354 | 12,21814 | 10,00733 | 6,75557 | 5,283253 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,592185 | 1,943929 | 2,87963 | 3,682115 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,796092 | 0,971964 | 0,719908 | 0,920529 |
| **Prędkość przetwarzania** | 1 542 135 | 2 455 365 | 2 997 802 | 4 440 780 | 5 678 320 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 40,18294 | 0,031557 | 100,7074 | 1,090332 |
| **Koszt zrównoleglenia** | BRAK | 4,909609 | 0,003158 | 6,803358 | 0,057605 |

*Tabela 2. Tabela przedstawiająca miary efektywnego przetwarzania dla przedziału liczb <2, 30 000 000> (2P - algorytm był uruchamiany na 2 procesach itp.; parametry przydziału static/dynamic domyślne).*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Sekwencyjny** | **Static 2P** | **Dynamic 2P** | **Static 4P** | **Dynamic 4P** |
| **Czas pracy procesorów** | 33,45201 | 32,94529 | 33,30458 | 34,35379 | 34,97794 |
| **Upływ czasu rzeczywistego** | 33,45325 | 17,64873 | 16,65324 | 9,47133 | 8,762171 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,895505 | 2,008814 | 3,532055 | 3,817918 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,947752 | 1,004407 | 0,883014 | 0,95448 |
| **Prędkość przetwarzania** | 896 773,8 | 1 699 839 | 1 801 452 | 3 167 454 | 3 423 809 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 13,32769 | 0,011343 | 37,28656 | 0,807391 |
| **Koszt zrównoleglenia** | BRAK | 2,352168 | 0,001889 | 3,531533 | 0,070745 |

*Tabela 3. Tabela przedstawiająca miary efektywnego przetwarzania dla przedziału liczb <30 000 000, 60 000 000> (2P - algorytm był uruchamiany na 2 procesach itp.; parametry przydziału static/dynamic domyślne).*

**b) sito domenowe:**

Algorytm był uruchamiany i testowany na wielkościach instancji, które zapewniały czas obliczeń poniżej 1 minuty oraz zakresach nie wyczerpujących pamięci: <2, 1 000 000 000> (2, MAX), <2, 500 000 000> (2, MAX/2), <500 000 000, 1 000 000 000> (MAX/2, MAX).

**S** - Algorytm sekwencyjny

**2\_StB, 4\_StB** - algorytm z przydziałem pracy statycznym blokowym uruchamianym na 2 lub 4 procesorach

**2\_C32, 4\_C256** - algorytm z przydziałem pracy statycznym cyklicznym i rozmiarze grupy równym "C*Liczba*" uruchamianym na 2 lub 4 procesorach

**2\_D1, 4\_D32** - algorytm z przydziałem pracy dynamicznym i rozmiarze grupy równym "D*Liczba*" uruchamianym na 2 lub 4 procesorach

- rozmiar <2, 1 000 000 000>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S** | **2\_StB** | **2\_C2** | **2\_C32** | **2\_C256** | **4\_StB** | **4\_C2** | **4\_C32** | **4\_C256** |
| **Czas pracy procesorów** | 14,73377 | 5,144037 | 5,543827 | 5,436262 | 5,390885 | 5,440801 | 5,696876 | 5,439055 | 5,416372 |
| **Upływ czasu rzeczywistego** | 14,73452 | 2,587081 | 2,828883 | 2,735183 | 2,716548 | 1,400293 | 1,772702 | 1,665981 | 1,595695 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 5,695423 | 5,2086 | 5,387033 | 5,423987 | 10,52246 | 8,3119 | 8,8443512 | 9,2339206 |
| **Efektywność przetwarzania równoległego** | BRAK | 2,847712 | 2,6043 | 2,693516 | 2,711993 | 2,630614 | 2,077975 | 2,2110878 | 2,3084802 |
| **Prędkość przetwarzania** | 6,78E+07 | 3,87E+08 | 3,53E+08 | 3,66E+08 | 3,68E+08 | 7,14E+08 | 5,64E+08 | 6,00E+08 | 6,26E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 1,16444 | 4,027703 | 1,246864 | 1,553847 | 11,45267 | 78,63318 | 73,522387 | 60,563454 |
| **Koszt zrównoleglenia** | BRAK | 0,030125 | 0,113939 | 0,034104 | 0,042211 | 0,160371 | 1,393932 | 1,224869 | 0,966408 |
|  | **S** | **2\_D1** | **2\_D2** | **2\_D32** | **2\_D256** | **4\_D1** | **4\_D2** | **4\_D32** | **4\_D256** |
| **Czas pracy procesorów** | 14,73377 | 5,233398 | 5,472978 | 5,272663 | 5,250039 | 5,440819 | 5,747567 | 5,283269 | 5,250228 |
| **Upływ czasu rzeczywistego** | 14,73452 | 2,621152 | 2,747804 | 2,641428 | 2,648443 | 1,365293 | 1,553719 | 1,398846 | 1,34827 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 5,621391 | 5,36229 | 5,578241 | 5,563465 | 10,7922 | 9,483389 | 10,53334 | 10,928465 |
| **Efektywność przetwarzania równoległego** | BRAK | 2,810696 | 2,681145 | 2,78912 | 2,781733 | 2,698051 | 2,370847 | 2,6333351 | 2,7321162 |
| **Prędkość przetwarzania** | 6,78E+07 | 3,82E+08 | 3,64E+08 | 3,79E+08 | 3,78E+08 | 7,32E+08 | 6,44E+08 | 7,14E+08 | 7,41E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 0,339774 | 0,823567 | 0,38589 | 1,768851 | 1,490742 | 30,0768 | 22,31232 | 10,595207 |
| **Koszt zrównoleglenia** | BRAK | 0,008906 | 0,02263 | 0,010193 | 0,046847 | 0,020353 | 0,467309 | 0,312115 | 0,142852 |

*Tabela 4. Tabela zbiorcza przedstawiająca miary efektywnego przetwarzania dla algorytmu sita domenowego i rozmiaru instancji <2, 1 000 000 000>.*

- rozmiar <2, 500 000 000>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S** | **2\_StB** | **2\_C2** | **2\_C32** | **2\_C256** | **4\_StB** | **4\_C2** | **4\_C32** | **4\_C256** |
| **Czas pracy procesorów** | 7,13402 | 2,595324 | 2,859829 | 2,731225 | 2,771916 | 2,729467 | 2,864973 | 2,732217 | 2,725558 |
| **Upływ czasu rzeczywistego** | 7,134413 | 1,307447 | 1,457263 | 1,373394 | 1,391788 | 0,687569 | 0,811628 | 0,758483 | 0,754792 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 5,4567512 | 4,8957621 | 5,1947314 | 5,1260774 | 10,376287 | 8,79025 | 9,4061607 | 9,4521577 |
| **Efektywność przetwarzania równoległego** | BRAK | 2,7283756 | 2,4478811 | 2,5973657 | 2,5630387 | 2,5940716 | 2,1975625 | 2,3515402 | 2,3630394 |
| **Prędkość przetwarzania** | 7,01E+07 | 3,82E+08 | 3,43E+08 | 3,64E+08 | 3,59E+08 | 7,27E+08 | 6,16E+08 | 6,59E+08 | 6,62E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 1,4968102 | 3,7534062 | 1,1331781 | 0,8377713 | 3,0264599 | 47,009098 | 39,778743 | 38,899458 |
| **Koszt zrównoleglenia** | BRAK | 0,01957 | 0,054697 | 0,015563 | 0,01166 | 0,020809 | 0,381539 | 0,301715 | 0,29361 |
|  | **S** | **2\_D1** | **2\_D2** | **2\_D32** | **2\_D256** | **4\_D1** | **4\_D2** | **4\_D32** | **4\_D256** |
| **Czas pracy procesorów** | 7,13402 | 2,613131 | 2,773035 | 2,672149 | 2,642043 | 2,750491 | 2,893373 | 2,662181 | 2,654076 |
| **Upływ czasu rzeczywistego** | 7,134413 | 1,306625 | 1,392771 | 1,340883 | 1,329587 | 0,690483 | 0,842596 | 0,714134 | 0,721597 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 5,4601841 | 5,1224595 | 5,3206827 | 5,3658865 | 10,332496 | 8,4671812 | 9,9903001 | 9,8869771 |
| **Efektywność przetwarzania równoległego** | BRAK | 2,730092 | 2,5612297 | 2,6603414 | 2,6829433 | 2,5831241 | 2,1167953 | 2,497575 | 2,4717443 |
| **Prędkość przetwarzania** | 7,01E+07 | 3,83E+08 | 3,59E+08 | 3,73E+08 | 3,76E+08 | 7,24E+08 | 5,93E+08 | 7,00E+08 | 6,93E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 0,0091074 | 0,897994 | 0,717214 | 1,2884452 | 1,6569561 | 56,612066 | 27,215481 | 32,194147 |
| **Koszt zrównoleglenia** | BRAK | 0,000119 | 0,012507 | 0,009617 | 0,017131 | 0,011441 | 0,477011 | 0,194355 | 0,232312 |

*Tabela 5. Tabela zbiorcza przedstawiająca miary efektywnego przetwarzania dla algorytmu sita domenowego i rozmiaru instancji <2, 500 000 000>.*

- rozmiar <500 000 000, 1 000 000 000>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S** | **2\_StB** | **2\_C2** | **2\_C32** | **2\_C256** | **4\_StB** | **4\_C2** | **4\_C32** | **4\_C256** |
| **Czas pracy procesorów** | 7,38376 | 3,033516 | 3,395949 | 3,144823 | 3,205257 | 3,190672 | 3,36415 | 3,18356 | 3,174215 |
| **Upływ czasu rzeczywistego** | 7,384134 | 1,519478 | 1,743034 | 1,581536 | 1,61103 | 0,801004 | 0,958114 | 0,879091 | 0,888609 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 4,8596518 | 4,2363683 | 4,6689636 | 4,5834863 | 9,2185982 | 7,7069472 | 8,3997379 | 8,3097673 |
| **Efektywność przetwarzania równoległego** | BRAK | 2,4298259 | 2,1181842 | 2,3344818 | 2,2917432 | 2,3046495 | 1,9267368 | 2,0999345 | 2,0774418 |
| **Prędkość przetwarzania** | 6,77E+07 | 3,29E+08 | 2,87E+08 | 3,16E+08 | 3,10E+08 | 6,24E+08 | 5,22E+08 | 5,69E+08 | 5,63E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 0,3580177 | 5,1702376 | 1,1538783 | 1,0429973 | 1,6659093 | 48,8779 | 37,857742 | 42,788335 |
| **Koszt zrównoleglenia** | BRAK | 0,00544 | 0,090119 | 0,018249 | 0,016803 | 0,013344 | 0,468306 | 0,332804 | 0,380221 |
|  | **S** | **2\_D1** | **2\_D2** | **2\_D32** | **2\_D256** | **4\_D1** | **4\_D2** | **4\_D32** | **4\_D256** |
| **Czas pracy procesorów** | 7,38376 | 3,030923 | 3,159471 | 3,120874 | 3,096546 | 3,198645 | 3,411324 | 3,10932 | 3,095338 |
| **Upływ czasu rzeczywistego** | 7,384134 | 1,517525 | 1,581557 | 1,566939 | 1,572363 | 0,803152 | 0,908291 | 0,807995 | 0,816895 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 4,865906 | 4,6689016 | 4,7124579 | 4,6962018 | 9,1939434 | 8,1297007 | 9,1388363 | 9,0392694 |
| **Efektywność przetwarzania równoległego** | BRAK | 2,432953 | 2,3344508 | 2,3562289 | 2,3481009 | 2,2984858 | 2,0324252 | 2,2847091 | 2,2598174 |
| **Prędkość przetwarzania** | 6,77E+07 | 3,29E+08 | 3,16E+08 | 3,19E+08 | 3,18E+08 | 6,23E+08 | 5,50E+08 | 6,19E+08 | 6,12E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 0,271956 | 0,2303426 | 0,8298983 | 3,0641779 | 1,7385252 | 24,423891 | 15,180787 | 21,084962 |
| **Koszt zrównoleglenia** | BRAK | 0,004127 | 0,003643 | 0,013004 | 0,04818 | 0,013963 | 0,22184 | 0,12266 | 0,172242 |

*Tabela 6. Tabela zbiorcza przedstawiająca miary efektywnego przetwarzania dla algorytmu sita domenowego i rozmiaru instancji <500 000 000, 1 000 000 000>.*

**c) sito funkcyjne:**

Algorytm był uruchamiany i testowany na wielkościach instancji, które zapewniały czas obliczeń poniżej 1 minuty oraz zakresach nie wyczerpujących pamięci: <2, 1 000 000 000> (2, MAX), <2, 500 000 000> (2, MAX/2), <500 000 000, 1 000 000 000> (MAX/2, MAX).

**S** - Algorytm sekwencyjny

**2\_StB, 4\_StB** - algorytm z przydziałem pracy statycznym blokowym uruchamianym na 2 lub 4 procesorach

**2\_C32, 4\_C256** - algorytm z przydziałem pracy statycznym cyklicznym i rozmiarze grupy równym "C*Liczba*" uruchamianym na 2 lub 4 procesorach

**2\_D1, 4\_D32** - algorytm z przydziałem pracy dynamicznym i rozmiarze grupy równym "D*Liczba*" uruchamianym na 2 lub 4 procesorach

- rozmiar <2, 1 000 000 000>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S** | **2\_StB** | **2\_C2** | **2\_C32** | **2\_C256** | **4\_StB** | **4\_C2** | **4\_C32** | **4\_C256** |
| **Czas pracy procesorów** | 14,73377 | 12,63317 | 19,968333 | 20,068194 | 21,105598 | 15,481476 | 27,786802 | 30,361781 | 26,717229 |
| **Upływ czasu rzeczywistego** | 14,73452 | 10,266452 | 10,127721 | 10,234808 | 11,592055 | 10,203986 | 9,894486 | 9,705063 | 9,921908 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,4352106 | 1,4548704 | 1,439648 | 1,2710879 | 1,4439966 | 1,4891649 | 1,5182303 | 1,4850491 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,7176053 | 0,7274352 | 0,719824 | 0,635544 | 0,3609991 | 0,3722912 | 0,3795576 | 0,3712623 |
| **Prędkość przetwarzania** | 6,79E+07 | 9,74E+07 | 9,87E+07 | 9,77E+07 | 8,63E+07 | 9,80E+07 | 1,01E+08 | 1,03E+08 | 1,01E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 76,94707 | 2,8348826 | 3,9221254 | 17,930488 | 248,28011 | 119,16882 | 87,155241 | 130,72489 |
| **Koszt zrównoleglenia** | BRAK | 7,899734 | 0,287109 | 0,401422 | 2,078512 | 25,334468 | 11,791142 | 8,458471 | 12,970403 |
|  | **S** | **2\_D1** | **2\_D2** | **2\_D32** | **2\_D256** | **4\_D1** | **4\_D2** | **4\_D32** | **4\_D256** |
| **Czas pracy procesorów** | 14,73377 | 16,630291 | 17,230035 | 17,710989 | 18,36291 | 26,941929 | 29,201197 | 30,831409 | 28,465933 |
| **Upływ czasu rzeczywistego** | 14,73452 | 8,324587 | 8,624875 | 8,859062 | 9,184322 | 6,899248 | 10,724892 | 10,927829 | 12,049661 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,7700002 | 1,708375 | 1,6632146 | 1,6043123 | 2,1356706 | 1,3738619 | 1,3483484 | 1,2228162 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,8850001 | 0,8541875 | 0,8316073 | 0,8021562 | 0,5339176 | 0,3434655 | 0,3370871 | 0,3057041 |
| **Prędkość przetwarzania** | 6,79E+07 | 1,20E+08 | 1,16E+08 | 1,13E+08 | 1,09E+08 | 1,45E+08 | 9,32E+07 | 9,15E+07 | 8,30E+07 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 0,2268341 | 0,228583 | 0,080539 | 0,0624325 | 9,4947015 | 127,72503 | 117,86337 | 163,76154 |
| **Koszt zrównoleglenia** | BRAK | 0,018883 | 0,019715 | 0,007135 | 0,005734 | 0,655063 | 13,698371 | 12,879907 | 19,732711 |

*Tabela 7. Tabela zbiorcza przedstawiająca miary efektywnego przetwarzania dla algorytmu sita funkcyjnego i rozmiaru instancji <2, 1 000 000 000>.*

- rozmiar <2, 500 000 000>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S** | **2\_StB** | **2\_C2** | **2\_C32** | **2\_C256** | **4\_StB** | **4\_C2** | **4\_C32** | **4\_C256** |
| **Czas pracy procesorów** | 7,13402 | 5,794507 | 9,794449 | 10,409504 | 11,013181 | 7,408863 | 14,609188 | 13,829385 | 12,091873 |
| **Upływ czasu rzeczywistego** | 7,134413 | 4,904947 | 4,959123 | 5,337033 | 6,270134 | 4,914389 | 4,029787 | 4,044826 | 4,754051 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,4545342 | 1,4386441 | 1,3367751 | 1,1378406 | 1,4517396 | 1,7704194 | 1,7638368 | 1,5007018 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,7272671 | 0,719322 | 0,6683876 | 0,5689203 | 0,3629349 | 0,4426048 | 0,4409592 | 0,3751755 |
| **Prędkość przetwarzania** | 7,01E+07 | 1,02E+08 | 1,01E+08 | 9,37E+07 | 7,97E+07 | 1,02E+08 | 1,24E+08 | 1,24E+08 | 1,05E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 81,864024 | 2,4963486 | 4,9570988 | 24,354934 | 249,24142 | 37,46997 | 58,096912 | 145,65117 |
| **Koszt zrównoleglenia** | BRAK | 4,015387 | 0,123797 | 0,264562 | 1,527087 | 12,248693 | 1,50996 | 2,349919 | 6,924331 |
|  | **S** | **2\_D1** | **2\_D2** | **2\_D32** | **2\_D256** | **4\_D1** | **4\_D2** | **4\_D32** | **4\_D256** |
| **Czas pracy procesorów** | 7,13402 | 7,220463 | 9,497274 | 10,243946 | 10,486446 | 12,765393 | 14,021482 | 14,607963 | 13,516276 |
| **Upływ czasu rzeczywistego** | 7,134413 | 3,612296 | 4,776052 | 5,149943 | 5,253124 | 3,251783 | 4,962826 | 5,233805 | 5,65424 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,9750355 | 1,4937888 | 1,3853382 | 1,3581277 | 2,1940003 | 1,4375707 | 1,3631408 | 1,2617811 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,9875178 | 0,7468944 | 0,6926691 | 0,6790638 | 0,5485001 | 0,3593927 | 0,3407852 | 0,3154453 |
| **Prędkość przetwarzania** | 7,01E+07 | 1,38E+08 | 1,05E+08 | 9,71E+07 | 9,52E+07 | 1,54E+08 | 1,01E+08 | 9,55E+07 | 8,84E+07 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 0,114304 | 1,1480193 | 1,0862256 | 0,3769566 | 7,4340446 | 117,4698 | 120,8921 | 160,95327 |
| **Koszt zrównoleglenia** | BRAK | 0,004129 | 0,05483 | 0,05594 | 0,019802 | 0,241739 | 5,829822 | 6,327257 | 9,100684 |

*Tabela 8. Tabela zbiorcza przedstawiająca miary efektywnego przetwarzania dla algorytmu sita funkcyjnego i rozmiaru instancji <2, 500 000 000>.*

- rozmiar <500 000 000, 1 000 000 000>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **S** | **2\_StB** | **2\_C2** | **2\_C32** | **2\_C256** | **4\_StB** | **4\_C2** | **4\_C32** | **4\_C256** |
| **Czas pracy procesorów** | 7,38376 | 5,952021 | 8,450882 | 10,681932 | 9,257012 | 7,478305 | 14,250754 | 14,380157 | 12,578775 |
| **Upływ czasu rzeczywistego** | 7,384134 | 5,094471 | 4,300227 | 5,467179 | 5,085198 | 5,090416 | 5,209076 | 4,131621 | 4,768768 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,4494408 | 1,7171498 | 1,3506296 | 1,4520839 | 1,4505954 | 1,4175516 | 1,7872244 | 1,5484364 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,7247204 | 0,8585749 | 0,6753148 | 0,7260419 | 0,3626488 | 0,3543879 | 0,4468061 | 0,3871091 |
| **Prędkość przetwarzania** | 6,77E+07 | 9,81E+07 | 1,16E+08 | 9,15E+07 | 9,83E+07 | 9,82E+07 | 9,60E+07 | 1,21E+08 | 1,05E+08 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 83,167045 | 3,478235 | 4,6171161 | 17,961621 | 253,09049 | 126,42453 | 51,948787 | 136,2259 |
| **Koszt zrównoleglenia** | BRAK | 4,236921 | 0,149572 | 0,252426 | 0,913384 | 12,883359 | 6,58555 | 2,146327 | 6,496297 |
|  | **S** | **2\_D1** | **2\_D2** | **2\_D32** | **2\_D256** | **4\_D1** | **4\_D2** | **4\_D32** | **4\_D256** |
| **Czas pracy procesorów** | 7,38376 | 7,514175 | 9,843238 | 8,748447 | 11,127268 | 13,071332 | 14,049202 | 15,116062 | 13,518476 |
| **Upływ czasu rzeczywistego** | 7,384134 | 3,758092 | 4,942438 | 4,37984 | 5,612501 | 3,287629 | 5,224325 | 5,368971 | 5,859131 |
| **Przyspieszenie przetwarzania równoległego** | BRAK | 1,9648625 | 1,4940266 | 1,6859369 | 1,3156584 | 2,2460363 | 1,413414 | 1,3753351 | 1,260278 |
| **Efektywność przetwarzania równoległego** | BRAK | 0,9824312 | 0,7470133 | 0,8429685 | 0,6578292 | 0,5615091 | 0,3533535 | 0,3438338 | 0,3150695 |
| **Prędkość przetwarzania** | 6,77E+07 | 1,33E+08 | 1,01E+08 | 1,14E+08 | 8,91E+07 | 1,52E+08 | 9,57E+07 | 9,31E+07 | 8,53E+07 |
| **Względny procentowy koszt zrównoleglenia** | BRAK | 0,053458 | 0,8424587 | 0,2564706 | 1,7413627 | 2,4085443 | 131,08101 | 118,45514 | 169,27507 |
| **Koszt zrównoleglenia** | BRAK | 0,002009 | 0,041638 | 0,011233 | 0,097734 | 0,079184 | 6,848098 | 6,359822 | 9,918048 |

*Tabela 9. Tabela zbiorcza przedstawiająca miary efektywnego przetwarzania dla algorytmu sita funkcyjnego i rozmiaru instancji <500 000 000, 1 000 000 000>.*

**4. Wnioski**

a) poprzez dzielenie:

Porównując wartości podstawowych miar efektywnego przetwarzania pomiędzy statycznym a dynamicznym przydziałem zadań do procesów można wysnuć wniosek, że dla tej wersji algorytmu zdecydowanie lepszym rozwiązaniem jest dynamiczny przydział zadań (na przykładzie rozmiaru instancji <2, 60 000 000>):

- przyspieszenie: 1.6 dla statycznego do 2.01 dla dynamicznego (2 procesory) oraz 2.95 do 3.8 (4 procesory): około 25-30% poprawa w obu przypadkach

- efektywność: 0.8 dla statycznego do 1 dla dynamicznego (2 procesory) oraz 0.74 do 0.95 (4 procesory): około 25-30% poprawa w obu przypadkach

- względny procentowy koszt zrównoleglenia: 40.825 dla statycznego do 0.007 dla dynamicznego (2 procesory) oraz 101.50 do 0.85 (4 procesory)

Podobna sytuacja występuje przy rozmiarze instancji <2, 30 000 000>. Jednakże przy rozmiarze <30 000 000, 60 000 000> miary zmieniają się:

- przyspieszenie: 1.9 dla statycznego do 2 dla dynamicznego (2 procesory) oraz 3.53 do 3.82 (4 procesory): około 5-8% poprawa w obu przypadkach

- efektywność: 0.95 dla statycznego do 1 dla dynamicznego (2 procesory) oraz 0.88 do 0.95 (4 procesory): około 5-8% poprawa w obu przypadkach

- względny procentowy koszt zrównoleglenia: 13.33 dla statycznego do 0.01 dla dynamicznego (2 procesory) oraz 37.29 do 0.81 (4 procesory)

Obserwowaną różnicę w wynikach testów można wyjaśnić tym, że im większa liczba, tym więcej należy sprawdzić jej dzielników. Dla rozmiarów instancji <2, MAX> oraz <2, MAX/2>, gdzie MAX równa się 60 000 000, w przydziale statycznym procesy o niższym id otrzymują mniejsze liczby i spędzają mniej czasu na ich przetwarzaniu, podczas gdy procesy o większym id nadal liczą. W przydziale dynamicznym w momencie zakończenia obliczeń przez proces zgłasza się on po następny zbiór liczb i nie ma przestojów w pracy procesorów. Ta różnica w działaniu niweluje się dla rozmiarów instancji <MAX/2, MAX>, gdzie każda liczba jest duża i posiada wiele dzielników.

b) sito domenowe i funkcyjne:

Jakość rozwiązań problemu:

Dla wszystkich typów algorytmów wartości domyślne przydziału zadań, to jest statyczny blokowy i dynamiczny o rozmiarze 1, okazywały się najlepsze pod względem przyspieszenia.

Dla algorytmu sita domenowego nie występuje diametralna różnica w przyspieszeniach pomiędzy strategiami przydziału zadań statycznym a dynamicznym, np. 5.7 dla statycznego blokowego a 5.62 dla dynamicznego (instancja <2, MAX>, gdzie MAX to 1 000 000 000) lub 4.86 dla statycznego blokowego a 4.87 dla dynamicznego (instancja <MAX/2, MAX>).

Dla algorytmu sita funkcyjnego obserwujemy już inne zjawisko (odpowiednio dla powyższych): 1.44 do 1.77 lub 1.45 do 1.96. Z kolei warto zauważyć, porównując te wyniki z powyższymi, że algorytm sita domenowego jest od 2.5 do 4 razy szybszy.

Efektywne wykorzystanie wewnętrznych struktur procesora:

Algorytm sita domenowego zdecydowanie lepiej wykorzystuje wewnętrzne struktury i działanie procesora, z powodu wcześniej wymienionych różnic pomiędzy dwoma podejściami (w opisie algorytmów/kodu i podejść): ładuje do pamięci podręcznej podtablicę raz i w ramach niej szuka wszystkich (jeden raz) liczb pierwszych.

Jakość zrównoleglenia przetwarzania (omówione na przykładzie rozmiaru instancji <2, 500 000 000>, <2, MAX/2>):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **2\_StB** | **2\_D1** | **4\_StB** | **4\_D1** |
| **Przyspieszenie** | 5,4567512 | 5,4601841 | 10,376287 | 10,332496 |
| **Efektywność** | 2,7283756 | 2,730092 | 2,5940716 | 2,5831241 |
| **Względny procentowy koszt zrównoleglenia** | 1,4968102 | 0,0091074 | 3,0264599 | 1,6569561 |

*Tabela 10. Tabela poglądowa miar efektywności przetwarzania równoległego dla algorytmu sita domenowego.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **2\_StB** | **2\_D1** | **4\_StB** | **4\_D1** |
| **Przyspieszenie** | 1,4545342 | 1,9750355 | 1,4517396 | 2,1940003 |
| **Efektywność** | 0,7272671 | 0,9875178 | 0,3629349 | 0,5485001 |
| **Względny procentowy koszt zrównoleglenia** | 81,864024 | 0,114304 | 249,24142 | 7,4340446 |

*Tabela 11. Tabela poglądowa miar efektywności przetwarzania równoległego dla algorytmu sita funkcyjnego.*

Efektywność algorytmu sita domenowego jest większa: 3.74 razy dla 2\_StB, 2.76 dla 2\_D1, 7.19 dla 4\_StB, 4.69 dla 4\_D1. Również pod względem kosztu zrównoleglenia algorytm domenowy jest lepszy (z powodu wyżej wymienionych różnic w interakcjach z pamięcią podręczną procesora). Ponadto można zauważyć, że dynamiczny przydział zadań do procesów wiąże się z niższym kosztem zrównoleglenia, np. 1.5 dla 2\_StB do 0.009 dla 2\_D1 lub 249.24 dla 4\_StB do 7.43 dla 4\_D1.

Podejście domenowe okazało się najlepsze. Ponadto jest to niezależne od rozmiaru instancji, sposobie przydziału zadań do procesów i tym podobne. Związane jest to z pojedynczym ładowaniem podtablicy do pamięci podręcznej procesora i wyszukiwaniem w niej wszystkich liczb pierwszych na raz.