|  |  |  |
| --- | --- | --- |
| Imię Nazwisko – nr albumu:  Anna Dybel - 148901  Michał Janeczko - 148905  Tomasz Seruga - 130818 | Przedmiot:  Programowanie równoległe i rozproszone | Data:  16.05.2023r. |

Cel: Implementacja w środowisku OpenMP, MPI oraz hybrydowym (MPI+OpenMP) algorytmu znajdowania liczb pierwszych – sito Eratostenesa.

Przebieg projektu:

1. Znajdowanie liczb pierwszych algorytmem sita Eratostenes

Podstawowym zadaniem algorytmu sita Eratostenes jest wyznaczenie liczb pierwszych z przedziału 2 .. n, dla pewnej liczby naturalnej n. Głównym założeniem algorytmu na którym się opiera jest fakt, że do wyznaczenia liczb pierwszych z przedziału B = [sqrtn] + 1 .. n wystarczy znajomość liczb pierwszych z przedziału A = 2 .. [sqrtn], ponieważ każda liczba złożona należąca do przedziału B dzieli się przez jedną lub więcej liczb z przedziału A. Wobec tego aby sprawdzić czy dowolna liczba należąca do przedziału B jest złożona, wystarczy sprawdzić czy dzieli się bez reszty przez jedną lub więcej liczb z przedziału A.

1. Wersja sekwencyjna algorytmu

Niezbędnym etapem projektu było zaimplementowanie sekwencyjnej wersji algorytmu. Sekwencyjna implementacja jest ważnym elementem oceny efektywności równoległych implementacji. Bardzo istotne jest to aby sekwencyjny program był dobrze napisany tzn wykonanie takiego programu było najlepsze. Kolejnym ważnym aspektem jest wersji sekwencyjnej jest też to, że jeśli wykonujemy program dla małego rozmiaru zadania to nie ma sensu uruchamiać go równolegle dlatego że takie podejście może nie przynieść żadnych zysków a wręcz przeciwnie wykonanie takie programu może okazać się gorszę od sekwencyjnego.

1. Implementacja równoległa algorytmu

Przed przystąpieniem do wykonania równoległych implementacji została zwrócona uwaga na to, że zadanie obliczeniowe można podzielić na mniejsze zadania poprzez wykorzystanie dekompozycji danych. Oznacza to że zbiór B, w którym szukane są pozostałe liczby pierwsze można podzielić na mniejsze podzbiory i każdy taki podzbiór przypisywać wątką/procesą które szukają w nich liczb pierwszych. Następnie otrzymane podzbiory liczb pierwszych przez poszczególne wątki/procesy składają się na cały zbiór szukanych liczb pierwszych. Koleją obserwacją jest to, że wykonanie części w której wyznaczane są liczby pierwsze w podzbiorze A może nie przynieść żadnych zysków czasowych ze względu na komunikację. Wobec tego ta część wykonywana jest sekwencyjnie.

* 1. Wersja dla środowiska OpenMP

OpenMP, jest wieloplatformowym interfejsem programowania umożliwiającym tworzenie równoległych programów dla systemów wieloprocesorowych z pamięcią współdzieloną. Dzięki temu że składa się on z dyrektyw kompilatora i zmiennych środowiskowych, nie jest wymagane tworzenie programu od zera. Wobec tego do utworzenia wersji dla OpenMP została wykorzystana wersja sekwencyjna, która została poszerzona o odpowiednie dyrektyw. Dodatkowo wersja OpenMP uruchomiona dla jednego wątku jest wykonywana sekwencyjnie.

* 1. Wersja dla środowiska MPI

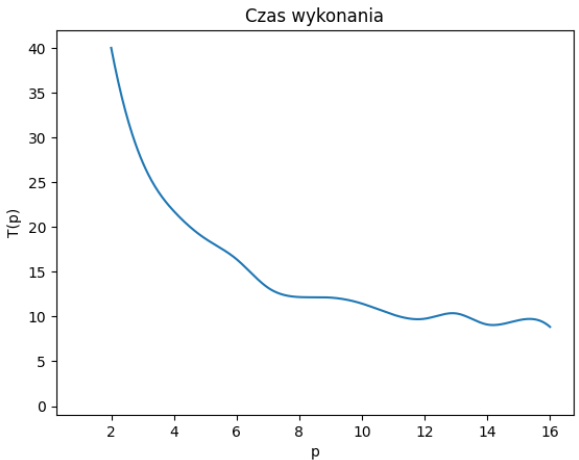
Kolejny punkt projektu polegał na wykonaniu implementacji algorytmu w środowisku MPI, który jest standardem przesyłania komunikatów pomiędzy procesami. Jego ogromnym atutem jest jego przenośność. Oznacza to że standard może być używany do programowania z modelem z pamięcią rozproszoną jak i programowania wielowątkowego z pamięcią wspólną. Jednak wykonanie wersji równoległej w standardzie MPI wymagała większego nakładu pracy ponieważ program musiał zostać napisać od nowa a nie tak jak w przypadku OpenMP rozszerzenie wersji sekwencyjnej o dyrektywy.

* 1. Wersja dla środowiska hybrydowego (OpenMP + MPI)

Standard OpenMP jest dedykowany do programowania z wielowątkowego z modelem pamięci wspólnej. Jednak może zostać także zastosowane wraz z MPI, tworząc hybrydową wersję dedykowaną klastrą komputerowym. W tym etapie wersja dla środowiska MPI została poszerzona o odpowiednie dyrektywy MPI.

1. Badanie efektywności obliczeń równoległych

Pomiary zostały wykonane dla programów o stałym rozmiarze zadania obliczeniowego przy zwiększaniu mocy obliczeniowej (liczbie procesorów). Ze względu na brak możliwość przetestowania wersji hybrydowej na sprzęcie docelowym czyli klastrach homogenicznych, uzyskane wyniki mogą nie przedstawiać rzeczywistych korzyści, które płyną z tej wersji implementacji równoległej. Wyniki zostały zaprezentowane w postaci wykresów i w formie tabeli.

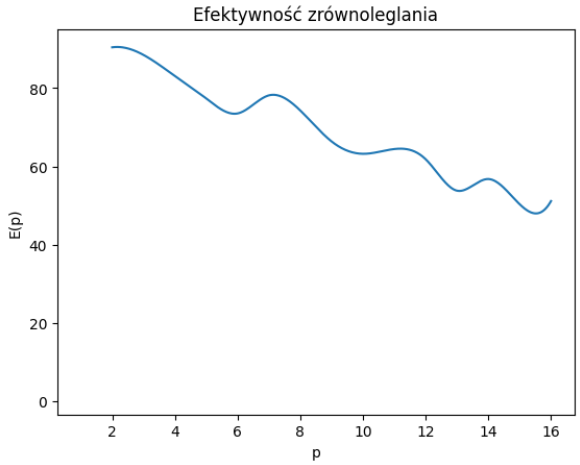


Wykres 1 Czas wykonania dla wersji OpenMP

Obraz zawierający tekst, diagram, Wykres, linia

Opis wygenerowany automatycznie

Wykres 2 Przyspieszenie obliczeń dla wersji OpenMP



Wykres 3 Efektywność zrównoleglania dla wersji OpenMP

Obraz zawierający linia, Wykres, diagram, tekst

Opis wygenerowany automatycznie

Wykres 4 Czas wykonania dla wersji MPI

Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie

Wykres 5 Przspieszenie obliczeń dla wersji MPI

Obraz zawierający tekst, linia, diagram, Wykres

Opis wygenerowany automatycznie

Wykres 6 Efektywność zrównoleglenia dla wersji MPI

Obraz zawierający tekst, diagram, linia, Wykres

Opis wygenerowany automatycznie

Wykres 7 Czas wykonania dla wersji hybrydowej

Obraz zawierający tekst, Wykres, diagram, linia

Opis wygenerowany automatycznie

Wykres 8 Przyspieszenie obliczeń dla wersji hybrydowe

Obraz zawierający tekst, Wykres, diagram, linia

Opis wygenerowany automatycznie

Wykres 9 Efektywność zrównoleglania dla wersji hybrydowej

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Implementacja | p | T | n | Speed up | Efficiency |
| hybrid | 2 | 42.762949 | 1E+08 | 1.6925788958100154 | 84.62894479050077 |
| hybrid | 3 | 15.715624 | 1E+08 | 4.60558645332823 | 153.51954844427433 |
| hybrid | 4 | 11.711095 | 1E+08 | 6.180435305152934 | 154.51088262882337 |
| hybrid | 5 | 9.685474 | 1E+08 | 7.473012162337126 | 149.46024324674252 |
| hybrid | 6 | 8.195771 | 1E+08 | 8.83134301824685 | 147.18905030411415 |
| hybrid | 7 | 7.406621 | 1E+08 | 9.772292250406764 | 139.6041750058109 |
| hybrid | 8 | 6.43834 | 1E+08 | 11.24197619262108 | 140.5247024077635 |
| hybrid | 9 | 5.886503 | 1E+08 | 12.29586819203184 | 136.62075768924268 |
| hybrid | 10 | 5.322901 | 1E+08 | 13.597785305418983 | 135.97785305418984 |
| hybrid | 11 | 5.090564 | 1E+08 | 14.218398000693048 | 129.25816364266407 |
| hybrid | 12 | 4.671822 | 1E+08 | 15.492813082347745 | 129.1067756862312 |
| hybrid | 13 | 4.378928 | 1E+08 | 16.52908314546391 | 127.14679342664547 |
| hybrid | 14 | 4.063362 | 1E+08 | 17.812753330862474 | 127.23395236330339 |
| hybrid | 15 | 3.965817 | 1E+08 | 18.250883739718702 | 121.67255826479135 |
| hybrid | 16 | 3.695422 | 1E+08 | 19.586305704734126 | 122.4144106545883 |
| mpi | 2 | 36.746874 | 1E+08 | 1.9696822374605254 | 98.48411187302627 |
| mpi | 3 | 25.880808 | 1E+08 | 2.796653991637356 | 93.22179972124519 |
| mpi | 4 | 22.039501 | 1E+08 | 3.284088192377858 | 82.10220480944645 |
| mpi | 5 | 17.838013 | 1E+08 | 4.0576080418822436 | 81.15216083764487 |
| mpi | 6 | 15.08173 | 1E+08 | 4.799161966167011 | 79.98603276945019 |
| mpi | 7 | 12.936471 | 1E+08 | 5.595008484153059 | 79.92869263075798 |
| mpi | 8 | 11.479943 | 1E+08 | 6.304880172314444 | 78.81100215393056 |
| mpi | 9 | 10.15851 | 1E+08 | 7.125027686146886 | 79.16697429052095 |
| mpi | 10 | 9.126191 | 1E+08 | 7.930982925954541 | 79.30982925954541 |
| mpi | 11 | 8.532759 | 1E+08 | 8.482562908433252 | 77.11420825848411 |
| mpi | 12 | 7.763797 | 1E+08 | 9.322714774742307 | 77.68928978951922 |
| mpi | 13 | 7.177007 | 1E+08 | 10.084937216864914 | 77.5764401297301 |
| mpi | 14 | 6.726539 | 1E+08 | 10.760312993056312 | 76.8593785218308 |
| mpi | 15 | 6.617141 | 1E+08 | 10.938208056923678 | 72.92138704615785 |
| mpi | 16 | 6.057528 | 1E+08 | 11.948713237479053 | 74.67945773424408 |
| openmp | 2 | 40.008658 | 1E+08 | 1.809100045295196 | 90.4550022647598 |
| openmp | 3 | 27.260211 | 1E+08 | 2.655139573204331 | 88.50465244014435 |
| openmp | 4 | 21.77001 | 1E+08 | 3.324741927082257 | 83.11854817705643 |
| openmp | 5 | 18.709427 | 1E+08 | 3.868620081202914 | 77.37240162405828 |
| openmp | 6 | 16.404498 | 1E+08 | 4.412184085121044 | 73.53640141868408 |
| openmp | 7 | 13.228338 | 1E+08 | 5.471561506819677 | 78.16516438313825 |
| openmp | 8 | 12.177388 | 1E+08 | 5.9437758737752295 | 74.29719842219036 |
| openmp | 9 | 12.116692 | 1E+08 | 5.973549959015217 | 66.3727773223913 |
| openmp | 10 | 11.449833 | 1E+08 | 6.321460321735697 | 63.21460321735697 |
| openmp | 11 | 10.212131 | 1E+08 | 7.087616189020686 | 64.4328744456426 |
| openmp | 12 | 9.754698 | 1E+08 | 7.419980095744636 | 61.83316746453863 |
| openmp | 13 | 10.352599 | 1E+08 | 6.9914487173703925 | 53.78037474900302 |
| openmp | 14 | 9.108507 | 1E+08 | 7.946380784468849 | 56.75986274620607 |
| openmp | 15 | 9.590617 | 1E+08 | 7.546924770324996 | 50.31283180216665 |
| openmp | 16 | 8.847957 | 1E+08 | 8.180381640643146 | 51.127385254019664 |

Podsumowanie:

Analizując uzyskane wykresy dla poszczególnych wersji można zauważyć, że im bardziej zwiększamy moc obliczeniową tym zyski są mniejsze i nie są już tak bardzo odczuwalne. Tego typu efekt jest spowodowany tym, że nie da się idealnie zrównoleglić algorytmu. Istnieje część sekwencyjna, która dla stałego rozmiaru zadania obliczeniowego zawsze będzie wykonywać się przez stały czas, którego nie można skrócić. Wobec tego program równoległy nie może osiągnąć czasu niższego niż czas wykonania części sekwencyjnej. Kolejnym wpływ mają także sekcje krytyczne, które zapobiegają wyścigu danych i zapewniają poprawne działania programu, wymiana informacja miedzy procesami, synchronizacja, zbieranie informacja przez proces od innych procesów. Wszystkie te czynniki wpływają na spadek wydajności wykonania równoległego. Biorąc pod uwagę otrzymane wyniki, najlepsze wyniki uzyskała implementacja hybrydowa, można tutaj zauważyć że wykres przyspieszenia dla tej implementacji jest superliniowy. Oznacza to, że dla danego p (mocy obliczeniowej, liczbie wykorzystanych procesorów) uzyskane przyspieszenie jest większe od p a z punktu teoretycznego idealne przyspieszenie powinno być równe p.

Jednak nie po to zwiększamy moc obliczeniową aby wykonywać to samo zadanie obliczeniowe, w tym przypadku o tym samym rozmiarze. Zwiększeniu mocy obliczeniowej powinno także towarzyszyć zwiększenie rozmiaru zadania lub jego złożoność, proporcjonalnie do zwiększanej mocy.