Programowanie równoległe. Przetwarzanie równoległe i rozproszone.

Laboratorium 9

Cel:
Nabycie umiejętności tworzenia i implementacji programów równoległych z wykorzystaniem OpenMP.

Kroki:

1. Utworzenie katalogu roboczego (np. *lab_9/petle*) .
2. Skopiowanie pliku *openmp_petle_simple.c*, analiza kodu, uruchomienie programu (kompilacja: *gcc -fopenmp openmp_petle_simple.c -o openmp_petle_simple)*
3. Zrównoleglenie wskazanej pętli za pomocą dyrektywy *parallel for*, w wersji domyślnej – bez klauzuli schedule, **z liczbą wątków sterowana przez wartość zmiennej środowiskowej OMP_NUM_THREADS**).
   - **Każdorazowo użycie klauzuli *default(none)* w celu wymuszenia jawnego ustalenia charakteru zmiennych (dotyczy to wszystkich programów zrównoleglanych w ramach zadań OpenMP)**
   - Sterowanie uzyskaniem ostatecznego wyniku w zmiennej *suma_parallel* powinno odbywać się **za pomocą klauzuli *reduction*** (jako podstawowego mechanizmu redukcji w OpenMP)
   - Obserwacja użycia **klauzuli o*rdered*** (należy ją dodać do odpowiednich dyrektyw) i dyrektywy *ordered* w celu wymuszenia kolejności wykonywania operacji (**służy to tylko celom ilustracji – w standardowych obliczeniach należy wymuszanie kolejności usunąć!)**
4. Uruchomienie dla czterech wątków (liczba iteracji 18 jest dobrana dla 4 wątków, tak aby wyraźnie pokazać mechanizmy przydziału iteracji wątkom), sprawdzenie poprawności działania (zgodność sumy), obserwacja wydruku (podczas tego laboratorium warto, jeśli jest taka możliwość, dostosować szerokość terminala w celu uzyskania dobrej czytelności wydruków)
5. Przetestowanie 4 wersji klauzuli *schedule* (za każdym razem **z czterema wątkami**, tym razem można użyć klauzuli *num_threads*) – w sprawozdaniu obrazy terminala z wydrukami można umieścić blisko siebie dla czytelnego zobrazowania różnic
   1. *static*, rozmiar porcji=3,
   2. *static*, rozmiar porcji domyślny,
   3. *dynamic* rozmiar porcji=2,
   4. *dynamic*, rozmiar porcji domyślny.

   Analiza wydruków – jaka jest domyślna porcja dla każdej z wersji?, jaka jest kolejność przydzielania iteracji wątkom?, która z wersji jest przyjmowana dla dyrektywy bez klauzuli *schedule*? czy wątki przy każdym uruchomieniu dostają te same iteracje? **(ocena)**
   **(odpowiedzi należy także umieścić w sprawozdaniu - 4 warianty, 4 wydruki ekranu, 4 omówienia)**
   *uwaga: przydział wszystkich iteracji jednemu wątkowi dla klauzuli schedule(dynamic...) jest możliwy, ale może świadczyć o wykonaniu nie w pełni równoległym (np. na jednym rdzeniu)*
6. Skopiowanie pliku *openmp_petle.c*, uruchomienie programu – **z liczbą wątków 3**.
7. Napisanie 4 wersji zrównoleglenia obliczania sumy wyrazów tablicy dwuwymiarowej – rozmiar tablicy należy dostosować tak, żeby ilustracje podziału iteracji między wątki były czytelne (jeden wiersz tablicy – jeden wiersz wydruku w terminalu, rozmiar tablicy minimum 10 – optymalny dla wydruków). W każdym przypadku należy utworzyć wydruki testowe – do pokazania i umieszczenia w sprawozdaniu. Wydruki powinny zawierać informację o przypisaniu każdej pozycji tablicy (czyli każdej iteracji) konkretnemu wątkowi. **Można zastosować wzór z pliku *openmp_petle.c*** (każdorazowo należy pozostawić indeksowanie tablicy a[i][j] i ewentualną zmianę kolejności przechodzenia tablicy realizować przez zmianę kolejności pętli). **Dla czytelności wydruku należy użyć klauzuli i dyrektywy *ordered*** oraz należy sprawdzić każdorazowo, że wersja równoległa daje te same wyniki co wersja sekwencyjna. Ocena powinna dotyczyć każdego kolejnego podpunktu, **dla każdego podpunktu** należy w sprawozdaniu umieścić fragment kodu, **zrzut ekranu z wykonania** i omówienie z odpowiedziami na wskazane pytania):

1. dekompozycja wierszowa – zrównoleglenie pętli zewnętrznej (pętla po wierszach jest pętlą zewnętrzną)
   - uzyskanie sumy poprzez klauzulę *reduction*
   - *schedule static*, rozmiar porcji=2,
   - obserwacja: całe obliczenia stanowią jeden obszar równoległy
   - do analizy: jakie przypisanie wierszy (czyli iteracji pętli zewnętrznej) wątkom daje zastosowany wariant klauzuli *schedule* ? (można posłużyć się analogią z tablicą 1D)
     - dobrą ilustracją dekompozycji (będącej w rzeczywistości dekompozycją danych) jest przedstawienie macierzy (tablicy 2D) jako prostokąta i pokazanie dekompozycji poprzez odpowiednie kolorowanie fragmentów prostokąta
2. dekompozycja kolumnowa - zrównoleglenie pętli wewnętrznej (pętla po kolumnach jest pętlą wewnętrzną)
   - uzyskanie sumy poprzez klauzulę *reduction*
   - *schedule dynamic*, rozmiar porcji domyślny,
   - obserwacja: każda iteracja sekwencyjnie wykonywanej pętli po wierszach (czyli dla każdego kolejnego wiersza) oznacza wejście w nowy obszar równoległy (jest tyle obszarów równoległych przy wykonywaniu kodu ile wierszy) – pętla zewnętrzna jest wykonywana przez jeden wątek (wątek główny procesu), dla każdego wiersza tworzone są nowe wątki (przynajmniej teoretycznie) , a po każdym wierszu następuje synchronizacja na końcu obszaru równoległego, połączona z redukcją zmiennej *suma_parallel*, która po obszarze równoległym jest standardową zmienną w wątku głównym
   - do analizy: jakie przypisanie kolumn (czyli iteracji pętli wewnętrznej) wątkom daje zastosowany wariant klauzuli *schedule* ? (można posłużyć się analogią z tablicą 1D)
     - dobrą ilustracją jest przedstawienie macierzy (tablicy 2D) jako prostokąta i pokazanie dekompozycji poprzez odpowiednie kolorowanie fragmentów prostokąta
     - czy zawsze podział kolumn pomiędzy wątki jest taki sam?
3. dekompozycja kolumnowa - zrównoleglenie pętli zewnętrznej (pętla po kolumnach jest pętlą zewnętrzną, a pętla po wierszach wewnętrzną, można to osiągnąć poprzez pozostawienie dostępu do tablicy w postaci a[i][j] i zamianę kolejności pętli – najpierw pętla (zewnętrzna) po *j*, a potem pętla (wewnętrzna) po *i*) - w celu realizacji zadania najlepiej skopiować istniejący kod podwójnej pętli, zakomentować dotychczasową wersję (z pętlą po wierszach jako zewnętrzną) i zmodyfikować skopiowany fragment, tak żeby pętla po kolumnach stała się zewnętrzna
   **(uwaga: dla pętli po kolumnach jako pętli zewnętrznej wydruk w terminalu odpowiada transpozycji macierzy – w wierszu wydruku znajduje się kolumna tablicy!);**
   - uzyskanie sumy poprzez ręczne sterowanie: każdy wątek sumuje w swojej zmiennej prywatnej – ostateczne uzyskanie odpowiedniej wartości zmiennej wspólnej odbywa się w sekcji krytycznej (ręczne sterowanie wymusza na programiście działania, które w OpenMP realizuje klauzula *reduction* – nieużywanie klauzuli *reduction* jest ogólnie niewskazane i w tym konkretnym przypadku służy tylko celom poglądowym)
     - przy realizacji zadania można posłużyć się wzorcem z wykładu
   - obserwacja: całe obliczenia stanowią jeden obszar równoległy, użycie ręcznego sterowania wymusza rozdzielenie obszaru dyrektywy *parallel* na dwie części: równoległą pętlę *for* i sekcję krytyczną (można ją także zrealizować za pomocą dyrektywy *atomic*)
   - *schedule static*, rozmiar porcji domyślny,
   - do analizy: jakie przypisanie kolumn (czyli iteracji pętli zewnętrznej) wątkom daje zastosowany wariant klauzuli *schedule* ? (można posłużyć się analogią z tablicą 1D)
     - dobrą ilustracją jest przedstawienie macierzy (tablicy 2D) jako prostokąta i pokazanie dekompozycji poprzez odpowiednie kolorowanie fragmentów prostokąta

---------- 3.0 ----------

4. dekompozycja 2D (zrównoleglenie obu pętli – po wierszach i po kolumnach):
   *(można ponownie uzyskać kod do pracy przez skopiowanie pierwotnej podwójnej pętli, z pętlą po wierszach jako pętlą zewnętrzną, i pracować stosując wykomentowywanie i odkomentowywanie)*

- należy zapewnić realizację zagnieżdżenia obszarów równoległych ( *omp_set_nested(1);* )
- każda pętla zrównoleglana własnym obszarem równoległym (*#pragma omp parallel for*), każdy wątek z pierwszego obszaru równoległego staje się wątkiem W_0 w drugim obszarze równoległym (co oznacza, że jego indeks się zmienia – poza wątkiem W_0 z pierwszego obszaru)
- należy odpowiednio sterować liczbą wątków (np. 3 w wierszach i 2 w kolumnach)
- przydział iteracji wykonać w sposób statyczny z rozmiarami porcji: 2 w kolumnach, 2 w wierszach (bloczki 2x2 jako elementarna jednostka przydziału)
- do analizy: jakie uzyskuje się w tym przypadku przypisanie bloków tablicy do wątków?
  - dobrą ilustracją jest przedstawienie macierzy (tablicy 2D) jako prostokąta i pokazanie dekompozycji poprzez odpowiednie kolorowanie fragmentów prostokąta

Uwaga: dla podziału blokowego i zagnieżdżonej równoległości numer wątku wygodnie jest przedstawiać jako parę (np. *(i,j)* ): *i* - numer wątku w zewnętrznym obszarze równoległym (przekazany do wewnętrznego obszaru poprzez zmienną *firstprivate)*, *j* - numer wątku w wewnętrznym obszarze (nadawany przez OpenMP i zwracany przez *omp_get_thread_num()*) (jak zwykle dla czytelności wydruku należy użyć klauzul i dyrektyw *ordered*).

5. rozważenie wariantu dekompozycji 2D bez ustalania rozmiaru porcji ( *schedule(static)* ) - jaka teraz otrzymywana jest dekompozycja tablicy
   - dobrą ilustracją jest przedstawienie macierzy (tablicy 2D) jako prostokąta i pokazanie dekompozycji poprzez odpowiednie kolorowanie fragmentów prostokąta

---------- 4.0 ----------

cd.

1. dekompozycja wierszowa – zrównoleglenie pętli wewnętrznej
   - uzyskanie sumy poprzez dwuetapową procedurę:
     - każdy wątek zapisuje wynik w sobie przypisanych elementach specjalnie zaalokowanej współdzielonej tablicy 1D
       - tablica może mieć wymiar równy liczbie wątków, każdy wątek liczy swoją sumę (sumę sobie przydzielonych wyrazów)
       - tablica może mieć wymiar równy liczbie wierszy, każdy wątek dodaje wyraz w sobie przypisanym wierszu do wyrazu tablicy, który w efekcie staje się sumą wyrazów w pojedynczym wierszu
     - sumowanie wartości w tablicy, w celu uzyskania ostatecznej sumy, odbywa się po wyjściu z obszaru równoległego (taki wzorzec redukcji jest alternatywą dla klauzuli *reduction* – w niektórych środowiskach programowania równoległego nie ma konstrukcji dla redukcji i wtedy trzeba zastosować jakiś wariant ręcznego sterowania – tak jak tutaj lub jak w p. 3)
   - obserwacja: każda iteracja pętli po kolumnach (dla każdej kolejnej kolumny) oznacza wejście w nowy obszar równoległy – pętla zewnętrzna (po kolumnach) jest wykonywana przez jeden wątek, w każdej kolumnie realizowana jest dekompozycja wierszowa, po której następuje synchronizacja
   - *schedule static* (rozmiar porcji 1) – czy dekompozycja wierszowa w każdej kolumnie może być inna? (jako w osobnym obszarze równoległym),
   - *schedule dynamic* (rozmiar porcji domyślny, czyli także 1) – czy dekompozycja wierszowa w każdej kolumnie może być inna? (jako w osobnym obszarze równoległym),
   - do analizy: jakie przypisanie wierszy (czyli iteracji pętli wewnętrznej) wątkom dają różne warianty klauzuli *schedule* ?

Warunki zaliczenia:

1. Obecność na zajęciach i wykonanie kroków 1-7.3
2. Oddanie sprawozdania o treści i formie zgodnej z regulaminem ćwiczeń laboratoryjnych - z opisem zrealizowanych zadań, kodem źródłowym programów i wydrukami dla wszystkich wariantów zrównoleglenia oraz wnioskami zawierającymi m.in. odpowiedzi na pytania w kolejnych punktach (wnioski można zamieszczać bezpośrednio w opisach realizacji kolejnych punktów i ewentualnie podsumować na zakończenie)